Embedded Linux Wiki

http://elinux.org



@泰晓科技 TinyLab.org

Table of Contents

> From: Embedded Linux Wiki (by CE Linux Forum)
> Team: eLinux.org Chinese Translation Group (by TinyLab.org)
> Attend: *Star/fork* eLinux github repo; *follow/SMS* @TinyLab.org; *See More* Translation Plan

Embedded Linux Wiki (elinux.org)

The original Embedded Linux wiki is built with MediaWiki and created by CE Linux Forum.

Based on the Mediawki version, this gitbook version is created by the eLinux.org Chinese Translation Group, the primary aim is to easier its translation to the other languages via internet cooperation and the byproduct is that this wiki is available as a printable and readable book now, people can download its pdf, epub and mobi formats.

- Project Homepage: http://www.tinylab.org/elinux
- Git Repository: https://github.com/tinyclub/elinux
- Online Gitbook: http://tinylab.gitbooks.io/elinux

Introduction

Welcome to the eLinux wiki!

The purpose of this wiki is to preserve and present information about the development and use of Linux in embedded systems as well as open source projects and tools for general embedded development. To use this wiki, click on one of the portal links left.

The main portals of the site take you to lists of resources or collections of information, you can use to tackle problems in the particular area referred to. For example, if you have a problem with boot up time of your embedded Linux system, click on Boot Time. You can also see a list of all the pages on this site.

Policies & Guidelines

Please read Policies & Guidelines for a detail list of site policies.

Errata

If you see something wrong, please change it. If you know something more about an issue, please add it. Please help to extend this wiki. Thank you!

License

Build HowTo and Tips

Build

To build this book, please use Markdown Lab.

Tips

Error: getaddrinfo ENOTFOUND elinux.org

```
$ gitbook pdf
info: loading book configuration....
warn: you should specify a gitbook version to use in your book.json, for example: 2.x.x
info: OK
info: >0 plugins loaded
info: Parsing multilingual book, with 1 lanuages
info: Preparing language book en
info: loading book configuration....OK
info: >0 plugins loaded
info: start generation with pdf generator
info: clean pdf generatorOK
info: start generation with pdf generator
info: clean pdf generatorOK
stream.js:94
      throw er; // Unhandled stream error in pipe.
            ^
Error: getaddrinfo ENOTFOUND elinux.org
    at errnoException (dns.js:44:10)
    at GetAddrInfoReqWrap.onlookup [as oncomplete] (dns.js:94:26)
```

- Solution

  ```
  $ ping elinux.org
  PING elinux.org (140.211.15.183) 56(84) bytes of data.

  $ sudo -s
  $ echo "140.211.15.183 elinux.org" >> /etc/hosts
  ```

Development Portals

Resources or information about embedded Linux development, from device drivers development to system optimization.

> From: eLinux.org

Firmware

Firmware is software that resides in non-volatile memory.

See the Wikipedia entry for firmware

One form of firmware is the Bootloader used to initialize the machine and load additional software, such as the Linux kernel.

Note that the Bootloader page has a list of oft-used Linux bootloaders.

Categories:

- Embedded Dictionary
- Firmware

> From: eLinux.org

Bootloader

Briefly, a bootloader is the first software program that runs when a computer starts. It is responsible for loading and transferring control to the operating system kernel software (such as the Hurd or the Linux). The kernel, in turn, initializes the rest of the operating system (e.g. GNU).

List of bootloaders

Legend: **?**:Unknown **-**:Not supported

Bootloader	ARM	BFIN	MIPS	PPC	SH	x86	remarks
APEX	y	-	?	?	?	?	
Barebox (U-Boot-v2)	y	y	y	y	(in progress)	y	allows networked setup, integrated editor and scripting
Blob	y	-	-	-	-	-	
CFE	-	-	y	-	-	-	for specific BroadCom chipsets
coreboot (*LinuxBIOS*) [1]	y	-	-	-	-	y	Conference talk by Peter Stuge at Embedded Linux Conference Europe 2008, video
Kexecboot	y	-	-	-	-	?	is a second-stage bootloader, consisting of the Linux kernel and a small gui
U-Boot	y	y	y	y	y	y	allows networked setup
Grub	?	-	?	y	-	y	
Lilo	-	-	-	-	-	y	x86 only, requires nasm to build
MicroMonitor	y	y	?	y	y	?	
PMON 2000	-	-	y	-	-	-	
Qi	y	-	-	-	-	-	Very fast, simple boot direct to Linux
RedBoot	y	?	y	y	y	y	allows networked setup
Syslinux	-	-	-	-	-	y	variants (including) isolinux are very flexible for booting x86
Yaboot	-	-	-	y	-	-	
YAMON	-	-	y	-	-	-	

A lot more exhaustive list is available at Wikipedia

List of legacy boot loaders

Please don't pick any of these for new designs!

- rrload (RidgeRun, used for older TI OMAP boards)
- pmon (used for Linux MIPS, including BroadCom wifi router boards like Linksys)

See also

- Bootloader_Security_Resources

Category:

- Bootloaders

> From: eLinux.org

Security

Contents

Introduction

This page has information about Security technologies for Embedded Linux.

Technology/Project pages

- Security Hardware Resources
- Bootloader Security Resources
- Mandatory Access Control Comparison

Security Systems in Linux Kernel

SELinux

- SELinux - This provides an implementation of the Flask Flux Advanced Security Kernel for Linux. SELinux started as a kernel patch which was presented by the NSA to kernel developers during the 2001 kernel summit. Feeback from this presentation started the LSM project, and the SELinux project helped define large parts of the LSM interface.
  - OLS 2008 paper: SELinux for Consumer Electronics Devices Nakamura & Sameshima, Hitachi Software Engineering.
  - ELC 2008 presentation: Embedded SELinux

Tomoyo

- TOMOYO Linux is a technology to improve Linux's security originally developed by NTT DATA CORPORATION, Japan. TOMOYO Linux was released on November, 11, 2005 as an open source software under the GPL. TOMOYO Linux is a mechanism called Secure OS, which can perform fine grained access control by breaking access permissions into parts like SELinux.
    - ELC2007 presentation
    - OLS2007 BoF slides

Tomoyo was incorporated into mainline Linux kernel as of version 2.6.28.

SMACK

- SMACK - Simple Mandatory Access Control Kernel. This is a lightweight implementation of MAC in the Linux kernel.
    - Home Page (very simple): http://schaufler-ca.com/
    - LWN.net article: http://lwn.net/Articles/244531/
    - CELF-commissioned white paper: SMACK for Digital TV by Embedded Alley (now Mentor Graphics)

SMACK was incorporated into the mainline Linux kernel as of version 2.6.25

Older Information (From CELF investigations prior to 2005)

Documents

- CELF 1.0 Security Specification: Security Spec_R2

Key Requirements and the Related Technologies

Where the technologies are defined as follows:

1. Umbrella
2. Linux Security Module (LSM) framework
3. PAX patch – (currently x86 only)
4. LOMAC
5. LIDS
6. Netfilter
7. digsig/bsign/elfsig
8. Trusted Computing Group (TCG)
9. TPE (included with LIDS)
10. PRAMFS
11. ACL file system extensions
12. Posix capabilities associated with files

Requirements	Technologies
Reliability	10
Secure/trusted boot	8
Access control	1, 4, 5, 11, 12
Buffer/stack protection	3
Intrusion detection	5, 8
Configurable security	1, 2, 4, 5, 7, 9(?), 11, 12
Authentication	1, 7
Signed binaries	1, 7
Trusted connection	IPSec, SSL already supported
Secure services	1, 4, 5, 7, 8
Firewall	6
API support for security hardware	8
Secure field upgradeability	9
Authentication	8

Of the listed technologies the CELF Security Working Group is studying or supporting the following:

- Umbrella
- PAX - only monitor for now
- LIDS
- Signed Binaries
  - Dig Sig (part of DSI project at http://disec.sourceforge.net/)
  - Bsign (a Debian project at http://packages.debian.org/squeeze/bsign)
  - ELFSign
- Linux API for TCG - pending CELF NPO status and liaison discussions
- TPE - as a part of LIDS
- ACL file system extensions - for those that CELF needs (PRAMFS, JFFS2). Also follow LKLM discussions and maybe do implementations
- POSIX capabilities associated with files

Resources

Security Frameworks

- The Linux Security Modules (LSM) project provides a lightweight, general-purpose framework for access control. Contemporary computing environments are increasingly hostile. Adding enhanced access control models to the kernel improves host security and can help a server survive malicious attacks. Security research has provided many types of enhanced access controls effective for different environments. The LSM framework allows access control models to be implemented as loadable kernel modules.

- Medusa DS9 Security Project is a project to enhance the security of Linux kernel, which implements the ZP Security Framework. The main goal of a project is to implement a framework for implementation of any security model (unlike other secure Linux kernel projects).

Medusa DS9 is used to increase Linux's security. It consists of two major parts, Linux kernel changes and the user-space daemon. Kernel changes do the monitoring of syscalls, filesystem actions, and processes, and they implement the communication protocol. The security daemon communicates with the kernel using the character device to send and

receive packets. It contains the whole logic and implements the concrete security policy. That means that Medusa can implement any model of data protection; it depends only on configuration file, which is in fact a program in the internal programming language, somewhat similar to C.

- Rule Set Based Access Control (RSBAC) is a flexible, powerful and fast open source access control framework for current Linux kernels, which has been in stable production use since January 2000 (version 1.0.9a). All development is independent of governments and big companies, and no existing access control code has been reused.

The standard package includes a range of access control models like MAC, RC, ACL (see below). Furthermore, the runtime registration facility (REG) makes it easy to implement your own access control model as a kernel module and get it registered at runtime.

The RSBAC framework is based on the Generalized Framework for Access Control (GFAC) by Abrams and La Padula. All security relevant system calls are extended by security enforcement code. This code calls the central decision component, which in turn calls all active decision modules and generates a combined decision. This decision is then enforced by the system call extensions.

Decisions are based on the type of access (request type), the access target and on the values of attributes attached to the subject calling and to the target to be accessed. Additional independent attributes can be used by individual modules, e.g. the privacy module (PM). All attributes are stored in fully protected directories, one on each mounted device. Thus changes to attributes require special system calls provided.

- TrustedBSD MAC Framework - Mandatory access controls extend discretionary access controls by allowing administrators to enforce additional security for all subjects (e.g. processes or sockets) and objects (e.g. sockets, file system objects, sysctl nodes) in the system. Development of those new access control models is facilitated by the development of a flexible kernel access control extension framework, the TrustedBSD MAC Framework. This permits new access control models to be introduced as kernel modules.

- Trusted Computing Group (TCG) - TCG defines a security architecture based on the hardware-based root of trust. This is a cost effective solution to establish Trusted Computing on various platforms. For some introductory information see Seiji Munetoh and Nicholas Szeto's presentation, TCGOverviewPDF, on the Tech Conference 2005Docs page. The Trusted Platform Module (TPM) is a security chip bound to the platform and a key component of this architecture. TCG has a Mobile Phone WG which has released a use cases document that is applicable to many generic CE devices in addition to the mobile phone -- MPWG User Cases

Security Components

- SELinux - This provides an implementation of the Flask Flux Advanced Security Kernel for Linux. SELinux started as a kernel patch which was presented by the NSA to kernel developers during the 2001 kernel summit. Feeback from this presentation started the LSM project, and the SELinux project helped define large parts of the LSM interface

- Apparmor - Apparmor is an application security tool designed to provide an easy-to-use security framework for your applications.

- The Linux Intrusion Defence System (LIDS) is a kernel patch and admin tools which enhances the kernel's security by implementing Mandatory Access Control (MAC). When it is in effect, chosen file access, all system network administration operations, any capability use, raw device, memory, and I/O access can be made impossible even for root. You can define which programs can access specific files. It uses and extends the system capabilities bounding set to control the whole system and adds some network and filesystem security features to the kernel to enhance the security. You can finely tune the security protections online, hide sensitive processes, receive security alerts through the network, and more. LIDS has two version trees, 1.2 and 2.2. LIDS 2.2 supports kernel 2.6. LIDS 1.2 supports kernel 2.4 and it provides new functions, Trusted Path Execution(TPE) and Trusted Domain Enforcement(TDE). These are useful to create a sandbox. LIDS is released under GPL.

- TOMOYO Linux is a technology to improve Linux's security originally developed by NTT DATA CORPORATION, Japan. TOMOYO Linux was released on November, 11, 2005 as an open source software under the GPL. TOMOYO Linux is a mechanism called Secure OS, which can perform fine grained access control by breaking access permissions into parts like SELinux.

- ELC2007 presentation
- OLS2007 BoF slides
- CELF Wiki

Tomoyo was incorporated into mainline Linux kernel as of version 2.6.28.

- SMACK - Simple Mandatory Access Control Kernel. This is a lightweight implementation of MAC in the Linux kernel.
  - Home Page (very simple): http://schaufler-ca.com/
  - LWN.net article: http://lwn.net/Articles/244531/
  - CELF-commissioned white paper: SMACK for Digital TV by Embedded Alley (now Mentor Graphics)

SMACK was incorporated into the mainline Linux kernel as of version 2.6.25

- Umbrella for handhelds implements a combination of process based mandatory access control (MAC) and authentication of files for Linux on top of the Linux Security Modules framework. The MAC scheme is enforced by a set of restrictions for each process.

  - Restrictions of resources
  - Restrictions of access to network interfaces
  - Restrictions on process creation and signaling
  - Signed files
- LOMAC is a dynamically-loadable security module for Free UNIX kernels that uses Low Water-Mark Mandatory Access Control (MAC) to protect the integrity of processes and data from viruses, Trojan horses, malicious remote users, and compromised network server daemons. LOMAC is designed for compatibility and ease of use - to be a form of MAC typical users can live with.

LOMAC is an attempt to produce a form of MAC integrity protection that typical users can live with. LOMAC implements a simple form of MAC integrity protection based on Biba's Low Water-Mark model in a Loadable Kernel Module (LKM). LOMAC provides useful integrity protection against viruses, Trojan horses, malicious remote users, and compromised network servers without any modifications to the kernel, applications, or their existing configurations. LOMAC is designed to be easy to use. Its default configuration is intended to provide useful protection without being adjusted for the specific users, servers, or other software present on the system. LOMAC may be used to harden currently-deployed systems simply by loading the LKM into the kernel shortly after boot time.

- The Enforcer is a Linux Security Module designed to improve integrity of a computer running Linux by ensuring no tampering of the filesystem. It can interact with TCPA hardware to provide higher levels of assurance for software and sensitive data.

- Janus is a security tool for sandboxing untrusted applications within a restricted execution environment. This can be used to limit the harm that can be caused by any successful compromise of the application. We have successfully used Janus to jail Apache, bind, and other programs within a limited sandbox without disturbing application behavior, and we continue to seek experience with using this approach in production environments.

- Domain and Type Enforcement (DTE) is a mandatory access control system which assigns types to files and domains to processes. Access from domains to other domains and from domains to types is enforced according to the DTE policy. The first implementation of this project closely followed the description by TIS in the papers titled A Domain and Type Enforcement Prototype and Confining Root Programs with Domain and Type Enforcement.

- The Realtime Linux Security Module (LSM) is a loadable extension for Linux 2.6 kernels. It selectively grants realtime permissions to specific user groups or applications.

- ACL support for Linux kernel - This linux kernel patch / user code combination allows supporting full access control lists (ACLs) for the Linux kernel.

- http://www.hu.grsecurity.net/ grsecurity (mirrors, original site was here) - is an innovative approach to security utilizing a multi-layered detection, prevention, and containment model. It is licensed under the GPL.

It offers among many other features:

-

- An intelligent and robust Role-Based Access Control (RBAC) system that can generate least privilege policies for your entire system with no configuration
- Change root (chroot) hardening
- /tmp race prevention
- Extensive auditing
- Prevention of entire classes of exploits related to address space bugs (from the PaX project)
- Additional randomness in the TCP/IP stack
- A restriction that allows a user to only view his/her processes
- Every security alert or audit contains the IP address of the person that caused the event

Security features

- NX patch - recent patch for kernel to prohibit execution of code on stack segment LKML discussion about NX patch

Other Resources

- Trusted Boot
  - Security Hardware Resources -- Security Hardware Resources
  - Bootloader Security Resources -- Bootloader Security Resources

Security Activities

- Trusted Computing Group
- Linux Security Modules

Mailing lists

- Linux Security Modules mailing list

Conferences

- Linux Conf Au Linux Security 2009 (miniconf)
  - January 21, 2009
- Usenix Security Symposium July 31 - August 4, 2006
  - proceedings
- Ottawa Linux Symposium (OLS) July 19 - 22, 2006 http://www.linuxsymposium.org/2006
  - OLS Proceedings

Security Articles

- TOMOYO Linux and pathname-based security [LWN.net] Apr 2008
- The Linux Journal Aug 2003
- ARM's Trust Zone for Security
- TPM-based Linux Run-time Attestation

Moved to:

- TPM-based Linux Run-time Attestation correct

Papers

- Experimenting with TCPA/TCG Hardware
- A Comparison of Publicly Available Tools for Dynamic Buffer Overflow Prevention
- SMACK for Digital TV

Sample and Opensource code

- A sample GPL TCPA Linux driver for Red Hat 8
- Linux TPM Device Driver
- TCG Software Stack (TSS) for Linux
- A NetBSD driver and some useful links can be found at Rick Wash's Trusted Computingpage.

Category:

- Security

> From: eLinux.org

Bootloader Security Resources

Contents

Overview

This page has security information about bootloaders.

Technology/Project pages

- Security
- Security Hardware Resources

Security Enhancements

There are two methods of booting

- Trusted/Authenticated Boot: just reporting
- Secure Boot: boot can be halted

Trusted Computing Group (TCG)

TCG is a hardware-based security solution not only for the PC platform, but also applicable for embedded devices. To understand the TCG, TCG Specification Architecture Overview is a good document.

Using the Trusted Platform Module(TPM) security chip and write-protected boot-code, we will be able to implement the Trusted Boot efficiently. Unfortunately, Many existing TPMs are designed for PC Platform, it requires LPC bus. Thus you have to use glue logic to use such TPM with your system. But, Atmel(R) has been released TPM chip, AT97SC3201S which has I2C/SMBus interface.

Open Source Projects/Mailing Lists

RedBoot/eCos

- RedBoot is a complete bootstrap environment for embedded systems. Based on the eCos. Following security enhancement was based on the RedBoot.
- High Robustness Bootloader for x86 Platform provide the capability of having signed program binary images.

U-Boot

Project site: u-boot

GRUB

GRUB is a bootloader for PC Platform. There are two patches to enable the TCG's Trusted Boot.

(In this case, the BIOS must support TCG Trusted Boot)

- University Bochum, Trusted Grub
- TrouSerS, GRUB TCG patch

GRUB provides a password feature, only administrator can start the interactive operations.

EtherBoot

EtherBoot is a software package for creating ROM images that can download code over an Ethernet network to be executed on an x86 computer. "SafeBootMode means any NBI image that's downloaded is checked whether it contains a valid digital signature and if not, the user is notified."

Other Resources

- W. A. Arbaugh , D. J. Farber , J. M. Smith, A secure and reliable bootstrap architecture, Proceedings of the 1997 IEEE Symposium on Security and Privacy, p.65, May 04-07, 1997
- Security Enhanced Bootloader for Operating Systems
- Technical Overview of Windows Vista - Secure Startup

Categories:

- Security
- Bootloader

> From: eLinux.org

Mandatory Access Control Comparison

Table Of Contents:

This page has information about Mandatory Access Control (MAC) solutions, which is of interest to CE Linux Forum members, because MAC provide strong access control for CE device which has rich resources to be managed.

Contents

Comparison of MAC solution

_	LIDS	TOMOYO	RSBAC	SELinux	A A
Security Model	MAC(inode), TPE(1.2),TDE(1.2)	MAC(path)	MAC, RC, ACL, FF, UM, PM, DAZ, JAIL	MAC(label), TE,RBAC,MLC,MCS	M
Type	LSM (2.6), patch (2.4)	patch	patch	LSM	L
Current version (2.6)	2.2.2 for 2.6.14 (LSM)	1.1.3 for 2.6.11-17	1.2.7 for 2.6.16	in mainline	2 (
Current version (2.4)	1.2.2 for 2.4.30	1.1.3 for 2.4.20 - 32	1.2.7 for 2.4.32	obsolete	?
Policy learn mode	/lids/lids.ini	CCS=0 /root/security/profile0.txt	/etc/selinux/config	rsbac_softmode	
disable option	lids=0			selinux=0	
Policy location	/etc/lids/	/root/security/	?	/etc/selinux	?
Distributions			Hardened Gentoo	Redhat, Fedora Core, Hardened Gentoo	C S
(by 3rd party)	Fedora core, Debian	Fedora core, Debian	Debian	Suse, Ubuntu	S

Benchmark

MEN WORKING

Hardware : Sharp Zaurus C860, CPU :XScale 400MHz, Memory : --MB, OS : Openzaurus 3.5.4.1 + OPIE 1.2

Sizing

Kernel 2.6.16 (linux-openzaurus-2.6.16-r40, Static build)

	Normal	LIDS	TOMOYO	RSBAC	SELinux
Kernel size (Image)	2487744	2554880	2541808	2974224	?
Kernel size (zImage)	1181660	1205324	1207288	1351432	?
image size overhead	0	67136	54064	486480	?
policy size	0				
memory consumption	0				

Lmbench

Processor, Process, Local communication latencies

	Normal	LIDS	TOMOYO	RSBAC	SELinux
null call	0.46	0.46	0.46		
null I/O	1.77	1.97 (11%)	1.77		
stat	12.7	15.7 (24%)	12.8 (1%)		
open/close	18.7	22.5 (20%)	59 (216%)		
select TCP	91.3	91.6	91.3		
sig inst	2.89	2.83 (-2%)	2.84 (-2%)		
sig hndl	7.58	7.66 (1%)	9.25 (22%)		
fork	3795	3808	3757 (-1%)		
execve	13000	13000	15000 (15%)		
sh	36000	37000 (3%)	41000 (14%)		
ctxsw	175	186.3 (7%)	177.2		
pipe	356.9	375.6 (5%)	358.1		
AF_UNIX	674	718 (7%)	723 (7%)		
UDP	747.5	776.3 (4%)	765.1 (2%)		
RPC/UDP	969.1	1013 (5%)	1193 (23%)		
TCP	957.3	1004 (5%)	964.6 (1%)		
RPC/TCP	1332	1380 (4%)	1353 (2%)		
TCP connect	2302	2379 (3%)	2357 (2%)		
0KB create	461	605.7 (31%)	669.8 (45%)		
0KB delete	232.5	267.1 (15%)	329.5 (42%)		
10KB create	5128.2	5234.6 (2%)	5235.6 (2%)		
10KB delete	298.8	349.8 (17%)	415.1 (39%)		
Mmap latency	-	-	-		
Prot Fault	1.72	1.71	0.61 (-64%)		
Page Fault	92	92	86 (-7%)		

Unixbench

	Normal	LIDS	TOMOYO	RSBAC	SELinux
execl	89.3 lps	84.6	59.5		
file read 1KB	53974.0 KBps	52176	53505		
file write 1KB	328.0 KBps	321	376		
file copy 1KB	288.0 KBps	199	311		
file read 256B	34766.0 KBps	33831	34742		
file write 256B	133.0 KBps	121	138		
file copy 256B	126.0 KBps	121	121		
file read 4KB	69148.0 KBps	67961	68851		
file write 4KB	1417.0 KBps	1417	1333		
file copy 4KB	1268.0 KBps	1237	1249		
pipe	112917.5 lps	108924	112137		
pipe switching	2655.4 lps	2559.6	2700		
process creation	272.9 lps	367.8	276.4		
system call	269446.2 lps	267748	268823.9		
shell scripts (1)	82.2 lpm	77.6	58.6		
shell scripts (8)	5.3 lpm	5.6	5.4		
shell scripts (16)	2.0 lpm	0	2		

Summary

	LIDS	TOMOYO	RSBAC	SELinux	App Armor
build (kenrel) (easy:5 - 1:hard)	4	4	3	5	?
build (userland) (easy:5 - 1:hard)	4	4	3	?	?
image size	2%	2%	15%	3%	?
performance					?
policy lean mode (good:5 - 1:poor)	4	5	?	3	?
symlink	by wrapper	support(alias)			?
filesystem JFFS2	ok	ok			ok?

Other resources

Access Control Comparison Table http://gentoo-wiki.com/Access_Control_Comparison_Table

Category:

- Security

> From: eLinux.org

Security Hardware Resources

This page has information about hardware based security enhancement, which is of interest to CE Linux Forum members

Contents

Technology/Project pages

- Security
- Bootloader Security Resources

Solutions

- Secure Flashing/Booting Support
- Secure Storage
- Cryptographic Accelerators
- FIPS Compliant True Hardware RNG
- Secure DMA Channels

Products

Security chips

TPM (Trusted Platform Module)

TPM Specifications

Security enhanced processors

ARM(R) TrustZone(R)

ARM Trustzone

OMAP M-Shied

TI White Paper

Intel(R) Wireless Trusted Platform

IBM White Paper

CELL

The Cell processor has an 'isolated' SPU runtime environment. CBE Architecture document

Open Source Projects/Mailing Lists

TCG/TPM

- Linux TPM Device Driver: Device driver to enable the TPM chip as described by specifications at http://www.trustedcomputinggroup.org. The TPM chip will enable you to use hardware to securely store and protect your keys and personal data. See also the TrouSerS project.

The TPM device drivers are already included in the mainline kernel and is split up into two parts:

-
    - the generic tpm driver module tpm.ko which handles all the common stuff
    - a vendor specific part tpm_\.ko

If you have a recent tpm module, it is quite likely that it follows the vendor independent TIS Protocol specified by the TCG, which should be preferred over the vendor modules. This module is called tpm_tis.ko

- TrouSers: An open-source TCG Software Stack implementation, created and released by IBM.

Categories:

- Security Hardware Resources
- Security

> From: eLinux.org

SELinux

SELinux -- Security Enhanced Linux

Contents

Current works about embedded SELinux

Many codes are submitted to Linux and userland community.

Linux kernel

2.6.18

Xattr support for jffs2

2.6.24

Reducing read/write overhead[1]

Reducing memory usage:[2]

Improving performance in AVC miss:[3]

2.6.25

Audit support for SH:[4]

SELinux userland

Reducing size of library: Merged to libselinux 2.0.35: [5]

BusyBox

Applets related to SELinux are merged to BusyBox in 1.8.0.

Support to assign domain to applets is merged to 1.8.0: [6]

Policy

SELinux Policy Editor will be helpful. See [7].

If you prefer fine grained configuration, Reference policy[8] is better.

Example of porting

Openmoko port

http://code.google.com/p/selinux-openmoko/

http://www.cse.psu.edu/~mhassan/openmoko_se/

Technical documents, presentations

- SELinux for Consumer Electronics Devices, Paper for Ottawa Linux Symposium 2008, paper and video.
- Example of porting to SH (Super H) was reported in CELF Jambolee #18: [9].
- Xattr port to jffs2 (Japanese), [10]

Remaining issues

Policy

xattr

logfs, yaffs, cramfs do not support xattr yet.

Size

Category:

- Security

From: eLinux.org

Tech Conference 2005Docs

Presenters, Demo-ers, Panelists and Participants: Thanks very much for your participation in CELF's 2005 Technical Conference. I really enjoyed the conference, and hope you did as well.

/\ **Presenters:** Please post your technical conference presentations on this page. (See Instructions below the table)

Here is an article on LinuxDevices with images of the demo posters: http://www.linuxdevices.com/articles/AT9266731500.html

Presentations

Person	Session Description	Presentation
Greg Ungerer, SnapGear	uCLinux	presentation (pdf)
Tohru Nojiri, Hitachi / Hirohisa Iijima, Lineo Solutions	Linux Kernel State Tracer (LKST) technology	presentation part1(pdf), presentation part2(pdf)
Todd Poynor, Monta Vista	Dynamic Power Management	presentation (pdf), notes
Philippe Robin, ARM	Developing and Optimizing Linux on ARM Cores	presentation (PDF)
Ed Plowman	OpenGL ES, Open VG and Open MAX	presentation (PDF)
Matt Mackall	Linux-tiny and Directions for Small Systems	presentation (Open Office) (pdf)
Erik Andersen	uClibc	presentation (pdf)
John Vugts & Jeroen Brouwer, Philips	UHAPI tutorial	Introduction (pdf), Tutorial (pdf)
Pieter van der Meulen, Philips	Audio, Video, Graphics WG discussion	(Acrobat: pdf), (Power Point: ppt), (Open Office: sxi)
Dongjun Shin, Samsung	Flash Memory WG discussion	(PDF)
Scott Preece, Motorola	Mobile Phone Profile WG - intro and working session	Plenary presentation,Technical Session Presentation,API Status presentation
Mark Orvek, Monta Vista	Power Management WG discussion - suspend to flash technology, etc.	not available
Manas Saksena, Time Sys	Real Time WG discussion	not available
Hiroo Suyama, NEC	System Size WG discussion	presentation (pdf)
Matsubara & Hagiwara & Hisao Munakata, Renesas	Graphics APIs for Linux (including DirectFB and 3D)	presentation_3D(ppt), presentation_DirectFB(ppt)
Nigel Cunningham, Cyclades	Linux 2.6 Power Management	presentation (Open Office), (pdf)

Emmanuel Fleury & Kristian S�rensen, Aalborg University	Umbrella Security Framework	not available
Stephen Johnson, Panasonic	Security WG discussion	presentation (pdf)
Dennis Oliver Kropp	DirectFB	Media:CELF_2005_DFB_Slides.pdf
Michael Hunold	Linux DVB API v4	pdf
Markku Ursin, Movial	Creating GTK+ based UI's for embedded devices	pdf
Manas Saksena, Time Sys	2.6 Realtime preemption patch	presentation (pdf)
Seiji Munetoh, IBM & Nicholas Szeto, Sony	Integrating TCG (Trusted Computing Group) technology in Linux	TCGOverview PDF
Tim Bird, Sony	Bootup Time WG discussion - current bootup time projects	presentation (pdf), (ppt)
Andrew Morton, Lead kernel developerHisao Munakata, RenesasMark Orvek, Monta VistaRuud Derwig, PhilipsModerated by Tim Bird, Sony	Panel - Improving Commercial CE Involvement in Linux	not available

Instructions

Here are the steps to follow (read them BEFORE clicking):

- PDF format is preferred. If you can convert your presentation to PDF, please do so.
- Please remove any messages in your presentation which say it is confidential
  - If you used a CELF template that has "CELF confidential" in the footer, please upload the presentation so we can remove the footer. Or you can remove it yourself by editing the footer on the slide "master" page.
- Please make sure your filename does not have any spaces in it
- Click on the AttachFile link at the bottom of the page
  - on the form, enter your presentation file name (or browse to it)
  - click on "Add link to page".
  - click on "Upload" (your file will be added as an attachment to the page)
  - click on "ShowText" to see the page again.

If you want to be extra nice, you can add the appropriate "Media:" line to your entry in the table. Do this by clicking on EditText at the bottom of the page and copying the text from the bottom of the page to your line (where it currently says "not available").

See the entry for Tim Bird (second to last in the table) for an example of the syntax required. Each table line is one long line, even though it may wrap in your browser (so remove all line feeds from your line before saving this page.) Don't worry - if you don't do this or mess something up I will come along later and fix up the table for you.

Thanks

Attachments

Below this line is a list of attachments:

Media:BTWG-Discussion-Plenary2005.ppt

Media:BTWG-Discussion-Plenary2005.pdf

Media:TCG_CELF_050125.pdf

Media:CELF_Plenary_Meeting_2005_demo_posters.pdf

Media:20050125c-CELF_TPod.pdf

Media:DPM-CELF-Plenary2005.pdf

Media:DPM-CELF-Plenary2005-Notes.txt

Media:SZWG_2005_Plenary.pdf

Media:uclinux.pdf

Media:celf_linux_dvb_v4.pdf

Media:uClibc_CELF.pdf

Media:AVGWG-20050126.pdf

Media:AVGWG-20050126.ppt

Media:20050126-gtk.pdf

Media:CELF-Technical-Meeting-2005-MPPWG-session.pdf

Media:Plenary-Meeting-2005-MPPWG-report.pdf

Media:APIstatus_report.pdf

Media:SECWG-Discussion-Plenary2005.pdf

Media:UHAPI-Forum-Introduction-CELF-20050126.pdf

Media:UHAPI-Forum-Technical-CELF-20050126.pdf

Media:Optimizing_Linux_ARM.pdf

Media:Khronos-CELF.pdf

Media:3DG050126rev12.ppt

Media:celf0501dfbexp_final.ppt

Media:Real-Time-Preemption-Patchset.pdf

Media:Linux_2.6_Power_Management.pdf

Media:linux-tiny.sxi

Media:linux-tiny.pdf

Media:FM-SIG-discussion.pdf

Category:

- Events

> From: eLinux.org

TomoyoLinux



Contents

Overview

TOMOYO Linux is a Mandatory Access Control (MAC) implementation for Linux that can be used to increase the security of a system, while also being useful purely as a system analysis tool. It was launched in March 2003 and is sponsored by NTT DATA Corporation, Japan.

For more information, please visit our project page.

http://tomoyo.sourceforge.jp/

Start, today!

- TOMOYO Linux LiveCD Tutorial for Ubuntu 10.04 (version 1.8)
- TOMOYO Linux LiveCD Tutorial for CentOS 5.5 (version 1.8)

Install

Mainline version (2.3)

- How to use TOMOYO Linux

Original hook version (version 1.8)

- http://tomoyo.sourceforge.jp/1.8/

Arch Linux

- TOMOYO Linux - ArchWiki
- Perfect Hideout: TOMOYO Linux - 5 tips to streamline your experience
- Perfect Hideout: Arch Linux with TOMOYO Linux MAC

MeeGo 1.1

- TOMOYO Linux on MeeGo
- TOMOYO Linux on MeeGo 1.1 handset

Android

- TOMOYO Linux on Android
- Celf Presentation
- Learning, Analyzing and Protecting Android with TOMOYO Linux (JLS2009)

TOMOYO Linux on LFS

For those "tough guys", TOMOYO Linux just runs fine on LFS. Find yourself and make your own version.

- TOMOYO CBLFS

TOMOYO Linux on CAT760

- TOMOYO Linux on CAT760

Browse the code

- kernel/security/tomoyo

Presentations

Highlights

- TOMOYO at Slideshare

2010-08-12 LinuxCon 2010

- Your First Guide to "secure Linux"

2009-10-27 Smartbook/Netbook/Mobile Application Conference Taipei 2009

- TOMOYO Linux on Android

2009-10-23 Japan Linux Symposium 2009

- Kernel Development: Drawing Lessons from "Mistakes"
- "Learning, Analyzing and Protecting Android with TOMOYO Linux"

2009-9-23 LinuxCon2009

- What Does It Mean Being a Project Manager in Enterprise (Enterprise Edition)
- What Does It Mean Being a Project Manager in Enterprise (Open Source Spirit Edition)

2009-6-12 CE Linux Forum Japan Technical Jamboree 28

- Part 1: TOMOYO Linux Introduction and Q&A (Japanese)
    - Video
- Part 2: TOMOYO Linux on Android (English)
    - Video

2009-5-22 CE Linux Forum Japan Technical Jamboree 27

- TOMOYO Linux on Android
- Video

2009-1-21 Linux Conf Au "Linux Security 2009 (miniconf)"

- TOMOYO Linux Overview
- Deep Inside TOMOYO Linux
- LWN.net article, "LCA: The security panel"

2008-11-21 FreedomHEC Taipei (Chinese)

- "Secure Linux" Primer
- TOMOYO Linux: pragmatic and manageable security for Linux

- photo

2008-11-13 PacSec 2008

- Behavior-based countermeasure against SSH Brute Force Attack

2008-7-25 Ottawa Linux Symposium 2008 BoF

- "MAC for Linux, Time to Glean" (pdf)
- HTML version

2008-7-9 Linux Foundation Japan #8 Symposium

- Agenda
- "Realities of Mainlining - case of the TOMOYO Linux project" (pdf)
- movie

2008-4-15 Embedded Linux Conference 2008

- "How to analyze your Linux's behavior with TOMOYO Linux" (program)

2008-2-24 FOSDEM2008 (Embedded Developer Room)

- Program
- "TOMOYO Linux for Secure Embedded"
- photo

- http://www.thinkit.co.jp/article/87/3/ (Japanese)

2007-11-29 PacSec 2007

- PacSec2007 Report
- TOMOYO Linux: A Practical Method to Understand and Protect Your Own Linux Box
- TOMOYO Linux: A Practical Method to Understand and Protect Your Own Linux Box (with demo)
- Handouts (bilingual)
- photo

2007-06-29 Ottawa Linux Symposium 2007

- TOMOYO Linux BoF
- photo

2007-04-18 CELF Worldwide Embedded Linux Conference 2007

- "TOMOYO Linux - A Lightweight and Manageable Security System for PC and Embedded Linux"
- TOMOYO Linux Tutorial

For the memory of OLS2007

- Memorial of OLS2007 BOF
- OLS2007 Photos
- 「熱い言葉に背中を押されて」 (in Japanese)
- 「海外での講演、そして新たなチャレンジへ」 (in Japanese)

Articles

- [A report from JLS LWN.net]
- Perfect Hideout: Arch Linux with TOMOYO Linux MAC
- Mandriva Linux 2010 released >> IT - Chuiko | Information Technology News
- Tomoyo GUI | Eugeni's blog
- Linux 2.6.30 Gets Faster Boot « PixelEstudios.com
- Why Linux security has failed (for the past 10 years) - Subreption Blog
- Kernel prepatch 2.6.30-rc1 | LWN.net
- TOMOYO Linux and pathname-based security | LWN.net
- TOMOYO Linux | KernelTrap

Readings

- 「初体験 TOMOYO Linux」 (in Japanese)
- The World of TOMOYO Linux (in Japanese)

Mainline

Activities

- At a glance
- 1st posting (13 Jun, 2007)
- 2nd posting (24 Aug, 2007)
- 3rd posting (2 Oct, 2007)
- 4th posting (11 Oct, 2007)
- 5th posting (17 Nov, 2007)
- TOMOYO Linux Security Goal (27 Dec, 2007) (thread)
- 6th posting (8 Jan, 2008)
- 7th posting (4 Apr, 2008)
- 8th posting (1 May, 2008)
- Introduce new LSM hooks where vfsmount is available (17, Sep, 2008)
- 9th posting (24 Sep, 2008)
- 10th posting (9 Oct, 2008)
- 11th posting (20 Oct, 2008)
- for -mm tree (4 Nov, 2008)
- 13th posting (21 Nov, 2008)
- 14th posting (1 Jan, 2009)
- 15th posting (5 Feb, 2009)

Forecast

- Linux Weather Forecast/security - The Linux Foundation

Mailing List

- English tomoyo-users-en@lists.sourceforge.jp (via GMANE)
- English tomoyo-dev-en@lists.sourceforge.jp (via GMANE)
- Japanese tomoyo-users@lists.sourceforge.jp (via GMANE)

Check for updates?

- freshmeat.net: Project details for TOMOYO Linux
- tomoyo-announce@lists.sourceforge.net (read only) (Created Aug 15, 2008)

Talk Annonymously?

TOMOYO: Forum: TOMOYO :: Open Discussion - SourceForge.JP

Contact

Project Manager: Toshiharu Harada (NTT DATA CORPORATION)

Category:

- Linux

> From: eLinux.org

Security Hardware Resources

(Redirected from TPM)

This page has information about hardware based security enhancement, which is of interest to CE Linux Forum members

Contents

Technology/Project pages

- Security
- Bootloader Security Resources

Solutions

- Secure Flashing/Booting Support
- Secure Storage
- Cryptographic Accelerators
- FIPS Compliant True Hardware RNG
- Secure DMA Channels

Products

Security chips

TPM (Trusted Platform Module)

TPM Specifications

Security enhanced processors

ARM(R) TrustZone(R)

ARM Trustzone

OMAP M-Shied

TI White Paper

Intel(R) Wireless Trusted Platform

IBM White Paper

CELL

The Cell processor has an 'isolated' SPU runtime environment. CBE Architecture document

Open Source Projects/Mailing Lists

TCG/TPM

- Linux TPM Device Driver: Device driver to enable the TPM chip as described by specifications at http://www.trustedcomputinggroup.org. The TPM chip will enable you to use hardware to securely store and protect your keys and personal data. See also the TrouSerS project.

The TPM device drivers are already included in the mainline kernel and is split up into two parts:

-
    - the generic tpm driver module tpm.ko which handles all the common stuff
    - a vendor specific part tpm_\.ko

If you have a recent tpm module, it is quite likely that it follows the vendor independent TIS Protocol specified by the TCG, which should be preferred over the vendor modules. This module is called tpm_tis.ko

- TrouSers: An open-source TCG Software Stack implementation, created and released by IBM.

Categories:

- Security Hardware Resources
- Security

> From: eLinux.org

Real Time

Contents

Introduction

This page has information about Real-time usage of Linux. Also this page has information about timing systems for Linux. This is of interest to CE Linux Forum members, because many consumer electronics products have realtime requirements (e.g. in the areas of multi-media presentation, or communications)

Real Time Wiki

- Please note that the primary source of information for Real Time Linux information is the new RTWiki.

Software Projects

- Realtime Preemption - Ingo Molnar's patchset to add realtime preemption to the 2.6 Linux kernel
- Kernel Timer Systems
  - Various new proposals for changing the kernel timing system
- Soft IRQ Threads - Technology to put SoftIRQs in threads so they can be preempted.
  - ***NOTE:** Soft IRQ threads are now (Oct 2007) incorporated into the Realtime Preemption patch*
- High Resolution Timers - A system to support timers with sub-jiffy resolution
- Variable Scheduling Timeouts
  - A system to support variable timeouts for periodic system activities (also known as Tickless)

Hardware Implementations

Ti AM18XX PRUSSv1

Ti_AM33XX_PRUSSv2

Tools for PRUSS

Documents

- Building Embedded Linux Systems, 2nd edition discusses the realtime preemption patch.
- CELF Realtime Specification (from 2004, so it's pretty old)
- Realtime Preemption presentation by Manas at the 2005 CELF Technical Conference - Media:Real-Time-Preemption-Patchset.pdf
- Realtime Testing Best Practices

    - a document to show recent testing results, and give hints for how different tests are conducted and what pitfalls to avoid.
- Real time in embedded Linux systems
- Using Real-Time Linux
    - Presentation by Klaas van Gends at the ELC 2008. The video is available
- Frank Rowand's series of talks
    - *Adventures in real-time performance tuning*
        - Part 1, slides and video ELCE 2008 version
        - Part 2, slides and video ELCE 2008 version
    - Musings On Analysis of Measurements of a Real-Time Workload slides ELC 2009 version and video
    - Real-Time Linux Failure slides ELC 2010 version and full HD video and 450x800 video
    - How Linux PREEMPT_RT Works slides ELCE 2011 version
- *Real-time vs real-fast, how to choose*, conference given by Paul E. McKenney at the Ottawa Linux Symposium 2008. Paper and video
- Paper: "Embedded GNU/Linux and Real-Time an executive summary", 2010 by Robert Berger
    - This papers, prepared for the Embedded World Conference 2010, compares different real-time approaches (including PREEMPT_RT and dual-kernel approaches).
    - The paper has an extensive list of references.
- Tutorial RT-Preempt Tutorial
- OSADL Realtime Page with realtime latency testing farm.

Further Open Source Projects

- Xenomai - Real-time development framework, closely cooperating with the Linux kernel. Among other features, it provides a migration path from various RTOSes like VxWorks, PSOS+, etc. to Linux based on so-called skins.

Category:

- Real Time

> From: eLinux.org

High Resolution Timers

Contents

Description

The objective of the high resolution timers project is to implement the POSIX 1003.1b Section 14 (Clocks and Timers) API in Linux. This includes support for high resolution timers - that is, timers with accuracy better than 1 jiffy.

When the project started, the POSIX clocks and timers APIs were not supported by Linux. Over time, the clocks and timers APIs have been adopted, and core infrastructure support for high resolution timers has been accepted into the mainline kernel (in 2.6.21). However, as of this writing, not all embedded platforms has support for high resolution timers, and even when support is present in the kernel code, it can be tricky to configure it for the kernel.

Rationale

Currently, timers in Linux are only supported at a resolution of 1 jiffy. The length of a jiffy is dependent on the value of HZ in the Linux kernel, and is 1 millisecond on i386 and some other platforms, and 10 milliseconds on most embedded platforms.

Higher resolution timers are needed to allow the system to wake up and process data at more accurate intervals.

Resources

Projects

hrtimers - Thomas Gleixner's patch

One project to support high resolution timers is Thomas Gleixner's hrtimers.

Thomas gave a presentation at the Ottawa Linux Symposium, July 2006, presenting the current status of hrtimers. The presentation is here: OLS hrtimers

As of July 2006, "generic clock sources" was accepted into Linus' mainline kernel tree (2.6.18-rc??). This means it should be appear in the mainline 2.6.18 kernel version, when that is available. hrtimers should soon follow, likely appearing in 2.6.19.

In February of 2006, James Perkins of WindRiver wrote:

---

ktimers has been obsoleted by hrtimers, and the core of hrtimers was merged and is present in Linus' 2.6.16-rc2. hrtimers is used as the base for itimers, nanosleep, and posix-timers. hrtimers are well-described by Jonathan Corbet at http://lwn.net/Articles/167897/

Since only the core of hrtimers is in 2.6.16-rc2, the hrtimers generally use the system timer as their tick source and run at HZ. John Stultz' generalized time source code has not yet been merged. Thomas Gleixner is maintaining his git tree and has graciously published patches at http://www.tglx.de/projects/hrtimers/ that include generalized clocksource, new timeofday patches, and get you the real "high resolution" timers for a subset of architectures.

High-res timers work is experimental and shifting and has been focusing on getting x86 working first, if this is adequate for you and you can use 2.6.16 kernels it's recommended, and let us all know of any problems or improvements. In contrast, the previous implementation that George Anzinger lead provides a fairly comprehensive set of functionality, back in the 2.6.8-2.6.10 era, but it isn't an active project at this time.

Note that the current HRT maintainers objected to this characterization.

HRT - Geoge Anzinger's patch

Prior to hrtimers, the main patch which provided high resolution timers was George Anzinger's patch.The official HRT site for this patch is at:

- high-res-timers

Downloads

Patch

- See Patch Archive
- Tom Rini has posted some patches for earlier 2.6 kernels at:
  - trini patches

Utility programs

How To Use

In order to use high resolution timers, you need to verify that the kernel has support for this feature for your target processor (and board). Also, you need to configure support for it in the Linux kernel.

Set CONFIG_HIGH_RES_TIMERS=y in your kernel config.

Compile your kernel and install it on your target board.

To use the Posix Timers API, see this online resource [1]

How to detect if your timer system supports high resolution

Here are several ways you can identify if your system supports high resolution timers.

- Examine kernel startup messages

Watch the kernel boot messages, or use `dmesg`. If the kernel successfully turns on the high resolution timer feature, it will print the message "Switched to high resolution mode on CPU0" (or something similar) during startup.

- Examine /proc/timer_list

  You can also examine the timer_list, and see whether specific clocks are listed as supporting high resolution. Here is a dump of /proc/timer_list on an OSK (ARM-based) development board, showing the clocks configured for high resolution.

  - cat /proc/timer_list

```
Timer List Version: v0.3
HRTIMER_MAX_CLOCK_BASES: 2
now at 294115539550 nsecs

cpu: 0
 clock 0:
  .index:      0
  .resolution: 1 nsecs
  .get_time:   ktime_get_real
  .offset:     0 nsecs
active timers:
 clock 1:
  .index:      1
  .resolution: 1 nsecs
  .get_time:   ktime_get
  .offset:     0 nsecs
active timers:
 #0: <c1e39e38>, tick_sched_timer, S:01, tick_nohz_restart_sched_tick, swapper/0
 # expires at 294117187500 nsecs [in 1647950 nsecs]
 #1: <c1e39e38>, it_real_fn, S:01, do_setitimer, syslogd/796
 # expires at 1207087219238 nsecs [in 912971679688 nsecs]
  .expires_next   : 294117187500 nsecs
  .hres_active    : 1
  .nr_events      : 1635
  .nohz_mode      : 2
  .idle_tick      : 294078125000 nsecs
  .tick_stopped   : 0
  .idle_jiffies   : 4294966537
  .idle_calls     : 2798
  .idle_sleeps    : 1031
  .idle_entrytime : 294105407714 nsecs
  .idle_sleeptime : 286135498094 nsecs
  .last_jiffies   : 4294966541
  .next_jiffies   : 4294966555
  .idle_expires   : 294179687500 nsecs
jiffies: 4294966542


Tick Device: mode:     1
Clock Event Device: 32k-timer
 max_delta_ns:   2147483647
 min_delta_ns:   30517
 mult:           140737
 shift:          32
 mode:           3
 next_event:     294117187500 nsecs
 set_next_event: omap_32k_timer_set_next_event
 set_mode:       omap_32k_timer_set_mode
 event_handler:  hrtimer_interrupt
```

Here are some things to check:

1. Check the resolution reported for your clocks. If your clock supports high resolution, it will have a .resolution value of 1 nsecs. If it does not, then it will have a .resolution value that equals the number of nanoseconds in a jiffy (usually 10000 nsecs, on embedded platforms).
2. Check the event_handler for the Tick Device. If the event handlers is 'hrtimer_interrupt' then the clock is set up for high resolution handling. If the event handler is 'tick_handle_periodic', then the device is set up for regular tick-based handling.
3. Check the list of timers, and see if the attribute .hres_active has a value of 1. If so, then the high resolution timer feature is active.
4. Run a test program

   You can run a small test program, and actually measure that the timers are returning in less than the period of a jiffy. If they are, this is the most definitive proof that your kernel supports high resolution timers. One example program you can try is cyclictest. Here is a sample command line which will test timers using nanosleep:

   - cyclictest -n -p 80 -i 500 -l 5000

   This does a test of clock_nanosleep, with priority 80, at 500 microsecond intervals, running the 5000 iterations of the test.

How to validate

See above with regard to cyclictest

Sample Results

[Examples of use with measurement of the effects.]

Case Study 1

Case Study 2

Status

- Status: implemented
- Architecture Support:

  (for each arch, one of: unknown, patches apply, compiles, runs, works, accepted)

  - i386: works
  - ARM: unknown
  - PPC: works
  - MIPS: unknown
  - SH: unknown

Future Work/Action Items

Here is a list of things that could be worked on for this feature:

- Documentation
- Testing

Old information (for 2.4 kernel)

The High Resolution Timers system allows a user space program to be wake up from a timer event with better accuracy, when using the POSIX timer APIs. Without this system, the best accuracy that can be obtained for timer events is 1 jiffy. This depends on the setting of HZ in the kernel. In the 2.4 kernel, HZ was set to 100, which means that the best accuracy you could get on a timer wakeup in user space was 10 milliseconds.

Put differently, if you asked for a timer event in 500 microseconds, you would wake up in 10 milliseconds (at least).

To support this feature on a particular board, you have to add a kernel driver that uses a timer on the system and supports the interface documented in: `include/linux/hrtime.h (in the CELF tree)` Additional documentation about this feature is available in `Documentation/high-res-timers/`

Patches for high-res timers were first presented at the time of kernel version 2.5.47, in November, 2002. See early patches

Category:

- Kernel

> From: eLinux.org

Kernel Timer Systems

Contents

Timer Wheel, Jiffies and HZ (or, the way it was)

The original kernel timer system (called the "timer wheel) was based on incrementing a kernel-internal value (jiffies) every timer interrupt. The timer interrupt becomes the default scheduling quamtum, and all other timers are based on jiffies. The timer interrupt rate (and jiffy increment rate) is defined by a compile-time constant called HZ. Different platforms use different values for HZ. Historically, the kernel used 100 as the value for HZ, yielding a jiffy interval of 10 ms. With 2.4, the HZ value for i386 was changed to 1000, yeilding a jiffy interval of 1 ms. Recently (2.6.13) the kernel changed HZ for i386 to

1. (1000 was deemed too high).

Ingo Molnar's explanation of timer wheel performance

Ingo Molnar did an in-depth explanation about the performance of the current "timer wheel" implementation of timers. This was part of a series of messages trying to justify the addition of ktimers (which have different characteristics).

It is possibly the best explanation of the timer wheel avaiable: See http://lkml.org/lkml/2005/10/19/46 and http://lwn.net/Articles/156329/

ktimers

Update: ktimers have been replaced by hrtimer framework also by Thomas Gleixner, only using a set of functions and datastructures in linux/ktime.h.

Material needs rework

A bunch of material in this section needs to be created or expanded to take into account the hrtimer system by Thomas Gleixner.

clock events

Basic LWN coverage on clockevents concepts

clocksource

Clocksource Documentation patch that didn't get accepted. Has some coverage of clock sources although care to be taken by going through patch responses.

Clocksource is also related or the same as the GTOD (Generic time of Day) work by John Stultz that hrtimer framework depends on (as mentioned on p.18 in the OLS 2006 slides).

Also refer to the kernel documentation on High resolution timers and dynamic ticks design notes for some notes on clock source.

Timer information

There are two /proc files that are very useful for gathering information about timers on your system.

/proc/timer_list

/proc/timer_list has information about the currently configured clocks and timers on the system. This is useful for debugging the current status of the timer system (especially while you are developing clockevent and clocksource support for your platform.)

You can tell if high resolution is configured for you machine by looking at a few different things:

For standard resolution (at jiffy resolution), a clock will have a value for it's '.resolution' field equal to the period of a jiffy. For embedded machines, where HZ is typically 100, this will be 10 milliseconds, or 10000000 (ten million) nanoseconds.

Also for standard resolution, the Clock Event Device will have an event handler of "tick_handle_periodic".

For high resolution, the resolution of the clock will be listed as 1 nanosecond (which is ridiculous, but serves as an indicator of essentially arbitrary precision.) Also, the Clock Event Device will have an event handler of "hrtimer_interrupt".

[need more info here - and this should probably be written up and put in Documentation/filesystems/proc.txt]

/proc/timer_stats

/proc/timer_stats is a file in the /proc pseudo file system which allows you to see information about the routines that are requesting timers of the Linux kernel. By cat'ing this file, you can see which routines are using lots of timers, and how frequently they are requesting them. This can be of interest to see

To use /proc/timer_stats, configure the kernel with support for the feature. That is, set CONFIG_TIMER_STATS=y in your .config. This is on the Kernel Hacking menu, with the prompt: "Collect kernel timers statistics"

Compile and install your kernel, and reboot your machine.

To activate the collection of stats (and reset the counters), do "echo 1 >/proc/timer_stats"

To stop collecting stats, do "echo 0 >/proc/timer_stats"

You can dump the statistics either while the collection system is running or stopped. To dump the stats, use 'cat /proc/timer_stats'. This shows the average events/sec at the end as well so you get a rough idea of system activity.

/proc/timer_stats fields (for version 0.1 of the format) are:

```
<count>,  <pid> <command>   <start_func> (<expire_func>)
```

Dynamic ticks

Tickless kernel, dynamic ticks or NO_HZ is a config option that enables a kernel to run without a regular timer tick. The timer tick is a timer interrupt that is usually generated HZ times per second, with the value of HZ being set at compile time and varying between around 100 to 1500. Running without a timer tick means the kernel does less work when idle and can potentially save power because it does not have to wake up regularly just to service the timer. The configuration option is CONFIG_NO_HZ and is set by Tickless System (Dynamic Ticks), on the Kernel Features configuration menu.

- See the Clockevents and dyntick LNW.net article

Testing

To tell if dynamic ticks is supported in your kernel you can:

Look in dmesg for a line like this one:

```
# dmesg | grep -i nohz
Switched to NOHz mode on CPU #0
```

Or look at the timer interrupts and compare to jiffies:

```
# cat /proc/interrupts | grep -i time
# sleep 10
# cat /proc/interrupts | grep -i time
```

Powertop

Powertop is a tool that parses the /proc/timer_stats output and gives a picture of what is causing wakeups on your system. Minimizing these wakeups should allow you to decrease power consumption in your device. Powertop was originally written for the x86 architecture but also works for embedded processors. However, in order to get a clean display from it, you will need an ncurses lib with wide character support.

Here's a poor-man's version of powertop:

```
# watch "cat /proc/timer_stats | sort -nr | head -n 20"
```

timer API

- interval timers
- posix timer API
- sleep, usleep and nanosleep

time API

- do_gettimeofday

High Resolution Timers

See High Resolution Timers, which describe sub-jiffy timers.

Old timer wheel/jiffy replacement proposals

Jun Sun's "tock" proposal

See http://linux.junsun.net/HRT/index.html

This systems replaces jiffies and xtime with tocks (arch-dependent), mtime (monotonic time) and wtime (wall time), and proposes a strategy for migrating to that.

John Stultz

In 2005, John Stultz proposed changes to the timers to use a 64-bit nanosecond value as the base. He did a presentation and BOF at OLS 2005. (It should be available online)

Timer Tick Thread - LKML July 2005

There was a very long thread about timers, jiffies, and related subjects in July of 2005 on the kernel mailing list.

The title was: "Re: [PATCH] i386: Selectable Frequency of the Timer Interrupt"

Linus said jiffies is not going away

- still need 32-bit counter, shouldn't be real-time value (too much overhead to calculate)
- high-res timers shouldn't be sub-HZ, but instead, HZ should be high and timer tick should not be 1:1 with HZ
  - in other words, have HZ be high (like 2K), have the timer interrupt fire off at some lower frequency, and increment jiffies by more than one on each interrupt.
  - rationale for this is to keep a single sub-system

Arjan had good points about coalescing low-res timers

- 3 use cases:
  - low res timeouts
  - high res timer for periodic absolute wakeup (wake up every 10 ms, whether last one was late or nt
  - high res timer for periodic relative wakeup (wake up 10 ms from now)

Category:

- Kernel

> From: eLinux.org

Realtime Preemption

Contents

Description

Overview

Realtime Preemption is (as of this writing 12/21/2004) a patch which tries to improve realtime performance of the Linux kernel.

Recent patches from Ingo include a (large) number of technologies for improving preemption and debugging preemption issues with the Linux kernel.

An overview of the technologies is as follows:

- voluntary preempt = a set of voluntary preemption points for the kernel, to improve normal scheduling latency (These changes basically
- BKL change to semaphore
- latency tracer

Voluntary Preempt

Overview:

- if it's on at compile time, it can be turned off at runtime with the command line: "voluntary-preemption=0" or "voluntary-preemption=off"
- Creates a new function might_resched(), which is used by might_sleep().
    - might_resched calls cond_resched() if voluntary preemption is on.
    - Adds might_sleep in several places.

Conversion of Spinlocks to Mutexes

According to Ingo Molnar, it's primary author, "the big change in this release is the addition of PREEMPT_REALTIME, which is a new implementation of a fully preemptible kernel model"

For a brief description of the overall technology, see: http://kerneltrap.org/node/3995?PHPSESSID=4bc02ae16e5a27308031f3cd664fd574

Briefly, the technology makes spinlocks and rwlocks preemptible by default.

- the patch auto-detects at compile-time the type of lock to use for a spinlock (mutex or original raw_spinlock)
- it uses a feature of gcc to manage this (reducing patch size)
- it uses native Linux semaphores for preemption
- it convert rwlocks to rw-semaphores
- apparently, about 90 locks are targetted for NON-conversion to preemptibility (that is, they are preserved as RAW_SPINLOCKS)

Ingo mentioned at one time that this was about 20% of the locks in his kernel configuration, implying that there were about 450 spinlocks present in the kernel in his configuration.

Ingo said this about how well this works on Un-processor (UP) systems versus SMP systems.

```
...and no matter how well UP works, to fix SMP one has to 'cover' all the
necessary locks first before fixing it, which (drastic) increase in raw
locks invalidates most of the UP efforts of getting rid of raw locks.
That's why i decided to go for SMP primarily - didnt see much point in
going for UP.
```

Normally, in UP the spinlocks are compiled away. When PREEMPT is turned on (without the new patch) these spinlocks are turned into markers for non-preemptible regions. When RT-PREEMPT is used,

people working on/interested in this stuff

- Ingo Molnar, Red Hat, voluntary preemption, Ingo real-time preemption
- Sven Dietrich, Monta Vista, MV real-time preemption
- Daniel Walker, Monta Vista, priority inheritance??
- John Cooper, Time Sys, ???
- Tim Bird, Sony, port to 2.6.10-native, port to PPC
- Scott Woods, Time Sys, IRQ threading??

people working on related stuff

- Bill Huey, Lynux Works??, mmlinux

miscellaneous comments

Comments regarding the scheduling of RT tasks

Ingo said (in this message):

---

note that my -RT patchset includes scheduler changes that implement "global RT scheduling" on SMP systems. Give it a go, it's at:

```
http://redhat.com/~mingo/realtime-preempt/
```

you have to enable CONFIG_PREEMPT_RT to active this feature. I've designed this code to not hurt non-RT scheduling, and i've optimized performance for the 'lightly loaded case' (which is the most common to occur on mainline-using systems).

A very short description of the design: there's a global 'RT overload counter' - which is zero and causes no overhead if there is at most 1 RT task in every runqueue. (i.e. at most 2 RT tasks on a 2-way system, at most 4 RT tasks on a 4-way system, etc.) If the system gets into 'RT overload' mode (e.g. the third RT task gets activated on a 2-way box), then the scheduler starts to balance the RT tasks agressively. Also, whenever an RT task is preempted on a CPU, or is woken up but cannot preempt a higher-prio RT task on a given CPU, then it's 'pushed' to other CPUs if possible. This design avoids global locking (it avoids a global runqueue), which simplifies things immensely. (I first tried a global runqueue for RT tasks but the complexity impact was much bigger.)

(note that these scheduler changes are resonably self-contained and do not depend on other parts of PREEMPT_RT, so in theory they could be added to mainline too, after some time - given lots of testing and broad agreement.)

---

comments regarding the hard parts of this work

Ingo says (at: http://groups-beta.google.com/group/linux.kernel/msg/cf036477d30ab736)

some of the harder stuff:

- the handling of per-CPU data structures (get_cpu_var())
- RCU and softirq data structures
- the handling of the IRQ flag

comments about the number of raw spinlocks needed

Ingo says (at: http://groups-beta.google.com/group/linux.kernel/msg/e63b2860d2e993dd)

Sven Dietrich sdietr...@mvista.com wrote:

> IMO the number of raw_spinlocks should be lower, I said teens before.
>
> Theoretically, it should only need to be around hardware registers and some memory maps and cache code, plus interrupt controller and other SMP-contended hardware.
>
> yeah, fully agreed. Right now the 90 locks i have means roughly 20% of all locking still happens as raw spinlocks.
>
> But, there is a 'correctness' *minimum* set of spinlocks that *must* be raw spinlocks - this i tried to map in the -T4 patch. The patch does run on SMP systems for example. (it was developed as an SMP kernel - in fact i never compiled it as UP :-|.) If code has per-CPU or preemption assumptions then there is no choice but to make it a raw spinlock, until those assumptions are fixed.

Rationale

This feature is intended to provide much better realtime scheduling response for a Linux system.

Resources

Projects

Various parties are working on ports: Time Sys and Monta Vista, in particular, seem to have made ports to PPC and ARM platforms.

Specifications

None that I'm aware of.

Online resources

The original announcement for voluntary-preemption:

- http://people.redhat.com/mingo/realtime-preempt/older/ANNOUNCE-voluntary

Here's some stuff by Jonathon Corbet:

- http://lwn.net/Articles/106010/
- http://lwn.net/Articles/107269/
- http://lwn.net/Articles/108216/
- http://lwn.net/Articles/129511/

There's a page of links about RT for audio at:

- http://www.affenbande.org/~tapas/wiki/index.php?Low%20latency%20for%20audio%20work%20on%20linux%202.6.x

A brief introduction of RT patch (Sorry, in Japanese only):

- http://www.atmarkit.co.jp/fembedded/rtos03/rtos03a.html

- Paper: "Embedded GNU/Linux and Real-Time an executive summary", 2010 by Robert Berger
    - This papers, prepared for the Embedded World Conference 2010, compares different real-time approaches (including RT-preempt and dual-kernel approaches).
    - The paper has an extensive list of references, which are very good.

Downloads

Patch

```
See http://redhat.com/~mingo/realtime-preempt/
```

Utility programs

[other programs, user-space, test, etc. related to this technology]

How To Use

- apply patch
- choose desired preemption level
- compile kernel

Configuration variables

The patch introduces (or modifies) the following configuration variables:

Variable	Purpose
ASM_SEMAPHORES	
BLOCKER	
CRITICAL_IRQSOFF_TIMING	
CRITICAL_PREEMPT_TIMING	
CRITICAL_TIMING	
FRAME_POINTER	
LATENCY_TIMING	
LATENCY_TRACE	
MCOUNT	
PREEMPT	
PREEMPT_BKL	
PREEMPT_DESKTOP	
PREEMPT_HARDIRQS	
PREEMPT_NONE	
PREEMPT_RT	
PREEMPT_SOFTIRQS	
PREEMPT_TRACE	
PREEMPT_VOLUNTARY	
RTC_HISTOGRAM	
RT_DEADLOCK_DETECT	
RWSEM_GENERIC_SPINLOCK	
RWSEM_XCHGADD_ALGORITHM	
SPINLOCK_BKL	
USE_FRAME_POINTER	
WAKEUP_TIMING	

- retrieved from patch with command:

```
grep "[+-]config " realtime-preempt-2.6.10-mm1-V0.7.34-01 | sed "s/[+-]config //" | sort | uniq
```

How to validate

[put references to test plans, scripts, methods, etc. here]

- use included trace feature, or
- use included latency overrun reporting mechanism
- Preemption_Instrumentation

Related projects

Monta Vista released a similar technology, which had the following features:

See http://groups-beta.google.com/group/linux.kernel/msg/7eeef031d9ec1446

These RT enhancements are an integration of features developed by others and some new MontaVista components:

- Voluntary Preemption by Ingo Molnar
- IRQ thread patches by Scott Wood and Ingo Molnar
- BKL mutex patch by Ingo Molnar (with MV extensions)
- PMutex from Germany's Universitaet der Bundeswehr, Munich
- MontaVista mutex abstraction layer replacing spinlocks with mutexes

Sample Results

[Examples of use with measurement of the effects.]

Case Study 1

- Linux RT Benchmarking Framework
  - http://www.opersys.com/lrtbf/
- Summary of dicussion in LKLM (sorry in Japanese)
  - http://japan.linux.com/kernel/05/07/25/2334226.shtml?topic=1
  - http://japan.linux.com/kernel/05/08/29/0817208.shtml?topic=1

Case Study 2

Trevor Woerner published some results in November 2005 regarding some latency measurements he have been recording on the 2.6.14 kernel with Ingo's patches.

See http://geek.vtnet.ca/embedded/LatencyTests/html/index.html

Case Study 3

Status

- Rt_Preempt_Subpatch_Table
- Status: [not started??]

  (one of: not started, researched, implemented, measured, documented, accepted)

- Architecture Support:

  (for each arch, one of: unknown, patches apply, compiles, runs, works, accepted)

  - i386: unknown
  - ARM: unknown
  - PPC: unknown
  - MIPS: unknown
  - SH: unknown

Future Work/Action Items

Here is a list of things that could be worked on for this feature:

- help with mainlining???
- perform testing on multiple platforms

- provide use cases for justification
- what else?
- break patch into manageable pieces - doesn't Ingo use any kind of patch management system???

people who expressed interest

Manas Saksena, Jon Masters, Takeharu Kato, Ralph Siemsen, Jyunji Kondo

Categories:

- Kernel
- Tips and Tricks

> From: eLinux.org

Realtime Testing Best Practices

Contents

Introduction

This page is intended to serve as a collecting point for presentations, documents, results, links and descriptions about testing Realtime performance of Linux systems. In the first section, please upload or place links to presentations or documentsion on the subject of RT testing for linux.

Terminology

This document uses the definitions for real time terminology found in: Real Time Terms

Test programs

RT Measurement programs

Here is a list of programs that have been used for realtime testing:

lpptest

- lpptest - included in the RT-preempt patch
  - It consists of a
    1. driver in the linux kernel, to toggle a bit on the parallel port, and watch for a response toggle back
    2. a user program to cause the measurement to happen
    3. a driver to respond to this toggling
- with the RT-preempt patch applied, see:
  - drivers/char/lpptest.c
  - scripts/testlpp.c
- For some other modifications, see http://www.ussg.iu.edu/hypermail/linux/kernel/0702.2/0342.html
  - remove dependency on TSC

This requires a separate machine to send the signal on the parallel port and receive the response. (Can this be run with a loopback cable? It seems like this would disturb the findings).

Are there any writeups of use of this test?

RealFeel

- RealFeel -
  - code at: http://brain.mcmaster.ca/~hahn/realfeel.c

This program is a very simple test of how well a periodic interrupt is processed. The program programs a periodic interrupt using /dev/rtc to fire at a fixed interval. The program measures the time duration from interrupt to interrupt, and compares this to the expected value for the duration. This simple program just prints a list of variances from the expected value, forever.

This program uses the TSC in user space for timestamps.

RealFeel (ETRI version rf-etri)

This program (latency.c) extends realfeel in several ways:

- it adds command line arguments to allow runtime control of most parameters
- it adds a histogram feature to dump the results to a histogram
  - it can do both linear and logarithmic histograms
- it locks the process pages in memory (very important)
- it changes the scheduling priority to SCHED_FIFO, at highest priority (very important)
- it adds conditional code to trigger output to a parallel port pin (for capture to an external probe or logic analyzer)
- it abstracts the routine to get the timestamp, with the function: getticks()
- it handles the interrupt signal and does a clean exit of the main loop (on user break?)
- it tracks min, max and average latency for whole run, and for every 1000 cycles of the loop
- it adds a timestamp to the /dev/rtc driver, and reads this as part of the rtc data
  - how is rtc timestamp used??

Cyclictest

- Cyclictest - See http://rt.wiki.kernel.org/index.php/Cyclictest

LRTB

- Linux Real-Time Benchmarking Framework - See http://www.opersys.com/lrtbf/
  - quickie overview at: http://groups.google.com/group/linux.kernel/msg/11860ef9e4263fa3?hl=en&

Hourglass

- Hourglass is a synthetic real-time application that can be used to learn how CPU scheduling in a general-purpose operating system works at microsecond and millisecond granularities
  - See: http://www.cs.utah.edu/~regehr/hourglass/

Woerner test

Trevor Woerner wrote an interesting test which received an interrupt on the serial port, and pushed data through several processes, before sending back out the serial port. This test requires an external machine for triggering the test and measuring the results.

See Trevor Woerner's latency tests

Senoner test

Benno Senoner has a latency test that simulates and audio workload. See http://www.gardena.net/benno/linux/audio/

Used (and extended??) by Takahashi Iwai - see http://www.alsa-project.org/~iwai/latencytest-0.5.6.tar.gz

Test Features Table

Feature	Rf-etri	Williams	LRTB
Is it platform specific (for target)?	yes - i386	no, but requires serial port on target	no, but requires parallel port on target
How is interrupt generated?	periodic timer programmed via /dev/rtc	data on serial port	data on parallel port
What does test measure?	interrupt and scheduling latency	end-to-end response latency	end-to-end response latency

Benchmarking programs

- see Benchmark Programs
- some to look into:
  - hackbench
  - lmbench
  - unixbench

Stress programs

- Ingo Molnar has a shell script which he calls dohell
  - good candidates seem to be:
    - find
    - du
    - ping
- Cache Calibrator - see RT-Preempt howto

Stress actions

Here are some things that will kill your RT performance:

- write the time of day to the CMOS of your RTC (see drivers/char/rtc.c - only by code inspection, no test yet)

- have a bus-master device do a long DMA on the bus
- get a page fault on your RT process (can be prevented with mlockall)
- get multiple TLB flushes on your RT code path (how to cause this??)
- get lots of instruction and data cache misses on your RT code path
  - how to cause this?
    - go down error paths in the RT case?
    - be ON a big error case when the RT event happens?
    - push your main RT code path and data sets out of cache with other work (in your RT process), prior to the next RT event?
    - access data in a very non-localized way on your RT code path

Test Hardware

- LRTB uses a 3-machine system:
  - target, host, and logger
  - target is the system under test
  - host is a control system, and it also collects the data
  - logger is a special machine used to cause interrupts on the target, and record the time it takes for the target to respond
  - Paulo Marqes offered to create custom hardware for the logger

Issues and Techniques

This is a list of issues and techniques for dealing with them, having to do with testing realtime performance in Linux.

ping flood isn't good as stress test

At one of the sessions at ELC 2007, Nicholas McGuire stated that a pingflood test is actually a poor test of RT performance, since it causes locality in the networking code rather than stressing the system.

Here is a list of issues that have to be dealt with:

- what tests are available on all platforms?
  - is special clock hardware or registers required for a test (e.g. realfeel, which only supports i386?)
  - does the program cross-compile?
  - Does generation of the test conditions perturb the test results?
  - Is special external hardware required?
  - How is the system stressed?
    - How to stress memory (cause cache-flushes and swapping)
    - How to stress bad code paths (long error paths, fault injection?)
- How is performance measured?

Using the LATENCY_TRACE option

Quote about latency-test from Ingo:

```
I'm seeing roughly half of that worst-case IRQ latency on similar
hardware (2GHz Athlon64), so i believe your system has some hardware
latency that masks the capabilities of the underlying RTOS. It would be
interesting to see IRQSOFF_TIMING + LATENCY_TRACE critical path
information from the -RT tree. Just enable those two options in the
.config (on the host side), and do:

        echo 0 > /proc/sys/kernel/preempt_max_latency

and the kernel will begin measuring and tracing worst-case latency
paths. Then put some load on the host when you see a 50+ usec latency
reported to the syslog, send me the /proc/latency_trace. It should be a
matter of a few minutes to capture this information.
```

Number of samples recommended

Ingo wrote:

```
also, i'm wondering why you tested with only 1,000,000 samples. I
routinely do 100,000,000 sample tests, and i did one overnight test with
more than 1 billion samples, and the latency difference is quite
significant between say 1,000,000 samples and 100,000,000 samples. All
you need to do is to increase the rate of interrupts generated by the
logger - e.g. my testbox can handle 80,000 irqs/sec with only 15% CPU
overhead.
```

Things to watch for in testing

Another note from Ingo - see here

- Note the bit about IRQ 7 - what's up with that?

```
> First things first, we want to report back that our setup is validated
> before we go onto this one. So we've modified LRTBF to do the
> busy-wait thing.

here's another bug in the way you are testing PREEMPT_RT irq latencies.
Right now you are doing this in lrtbf-0.1a/drivers/par-test.c:

    if (request_irq ( PAR_TEST_IRQ,
                                          &par_test_irq_handler,
#if CONFIG_PREEMPT_RT
                                           SA_NODELAY,
#else //!CONFIG_PREEMPT_RT
                                           SA_INTERRUPT,
#endif //PREEMPT_RT

you should set the SA_INTERRUPT flag in the PREEMPT_RT case too! I.e.
the relevant line above should be:

                                           SA_NODELAY | SA_INTERRUPT,

otherwise par_test_irq_handler will run with interrupts enabled, opening
the window for other interrupts to be injected and increasing the
worst-case latency! Take a look at drivers/char/lpptest.c how to do this
properly. Also, double-check that there is no IRQ 7 thread running on
the PREEMPT_RT kernel, to make sure you are measuring irq latencies.
```

Tests results taxonomy

Test Table

Person	Company	Hardware	Kernel	test method	Measurement method	Results
Sangbae Lee	Samsung	OSK - OMAP (ARM) 192 MHZ)	2.4.20 and 2.6.10	using two machine test	ZI instrumentation - measure interrupt reponse latency	2.4.20 - 30~35us, 2.6.10 - 30~35us max
Sangbae Lee	Samsung	MIPS 264 MHZ	2.6.10 ??	??	??	??
Katsuya Matsubara	IGEL	SH4	2.6.??	??	??	??
YungJoon Jung	ETRI	Via Nehemiah (i386)	2.6.12	periodic interrupt	rf-etri - measure scheduling latency minus interrupt latency	30 us max scheduling latency with RT-preempt
Tsutomu Owa	Toshiba	Cell (ppc64)	2.6.12	??	??	??

Test presentations and documents

Presentations

[Add links here, most recent at top]

- [TBD Linux Kernel's performance comparison] by HyoJun Im of LG at RTWG 2nd Face-to-Face Meeting in Korea
  - Tested Linux Kernel's performance by using opensource benchmarks
  - Test Target : Intel Pentium 4 2GHz, 1GB Memory
    - compared among linux kernel 2.4.22, 2.6.18 with preemption patch, 2.6.23-rc
    - used opensource benchmark test programs
      - Realfeel : measure interrupt latency time
        - 2.6(preemption) - 3.3ms, 2.6(rc3) - 3.5ms, 2.4 - 258.9ms
      - Hackbench Test : measure the scalability of scheduler
        - TBD
      - InterBench Test : measure the latency time of interactive task under load
        - He has good result table to see
      - LMbench Test : measure context switching latency time
        - TBD
- Analysis of Interrupt Entry Latency in Linux 2.4 vs 2.6 by !SangBae Lee of Samsung for ELC 2007
  - Analyzed MV 3.1 (2.4.20) and MV 4.0 (2.6.10), using LTT, on OSK board (OMAP 5920 ARM 192 MHZ)
    - This is not a realtime-preempt patch applied test. Only tested between 2.4.20 and 2.6.10 kernel
  - Initial results were that linux.2.4.20 was 3X fast for best-case interrupt latency by using LTT
    - This test's problem was to use LTT, LTT had really high overhead for this test's comparing
  - After reviewing code and finding that the interrupt code path was almost identical, a different, more lightweight tracer was used (Zoom-in tracer) showing latencies were almost the same between 2.4 kernel and 2.6 kernel
    - This ZI instrumentation has low overhead, so it is suitable for interrupt reponse time measurement. It was written by SangBae Lee
  - Also measured on MIPS 264 MHZ (for real TV system), but following data was measured on OSK board
    - Interrupt response time measured:
      - with LTT instrumentation:
        - 2.6.10 - min = 30 us, max = 400 us
        - 2.4.20 - min = 10 us, max = 30 us
      - with ZI instrumentation:
        - 2.6.10 - min = 3 us, max = 30~35 us

    - 2.4.20 - min = 3 us, max = 30~35 us
  - Basic result = Don't use LTT for measuring RT performance
- Porting and Evaluating the Linux Realtime Preemption on Embedded Platform by Katsuya Matsubara of Igel at ELC 2007
- Realtime Preempt Patch Adaptation Experience (and Real Time BOF notes)
  - !YungJoon Jung of ETRI at ELC 2007
  - This is the presentation of Realtime BoF in ELC 2007. It includes realtime preempt patch adaptation kernel's test
  - Test on VIA Nehemiah board, 1GHZ, 256M memory
  - See http://tree.celinuxforum.org/CelfPriWiki/RealTime_20Performance_20Test (need to make this public)
  - has good charts comparing vanilla, voluntary preempt, preemptible kernel and RT-preempt
  - min = 5.6 us, max = 41.1 us
  - showed RT-preempt has throughput problems (reported by hackbench)
- Performance Measurement of PPC64 RT patch (update) (english text)
  - by Tsutomu Owa of Toshiba at CELF Jamboree 13
- Porting pre-empt RT patch on SuperH (english text)
  - by Katsuya Matsubara (IGEL) at CELF Jamboree 13
- Performance Measurement of PPC64 RT Patch (english text)
  - by Tsutomu Owa of Toshiba at CELF Jamboree 12
- Linux Realtime Preemption and Its Impact on ULDD by Katsuya Matsubara & Hitomi Takahashi of IGEL, for CELF Jamboree 11
  - very good summary of RT-preempt patch. Also good description of work done on SH4 and work on User Level device drivers.
  - Describes basic steps to do a new port of RT-preempt
- Experience with Realtime Performance
  - by Shinichi Ochiai of Mitsubishi Electric Corporation at CELF ELC 2006
  - This describes RT features and how they evolved from 2.4.20 to 2.6.16. Test results are shown for preemptible kernel (2.4.20), voluntary preemption, RT-preempt, and hybrid kernel approach (RTAI). The platforms tested were an SH4 board and an EDEN board, with a VIA processor (i386 clone). RT-preempt is shown to have good RT characteristics, for later kernel versions.
- PREEMPT-RT vs I-PIPE: the numbers, take 3
  - by Kristian Benoit, LKML message, 2005
  - about extensive testing by Kristian Benoit and Karim Yaghmour
  - See also PREEMPT RT vs ADEOS: the numbers, part 1
  - and PREEMPT_RT vs I-PIPE: the numbers, take 2
- Trevor Woerner's latency tests
  - Interesting host/target test of latency via transmission and reception of strings over serial port
- Real-Time Preemption Patchset
  - by Manas Saksena, CELF tech conference 2005
  - Good paper with overview of RT-preempt patch features
- Audio Latency on Linux Kernels
  - Takahashi Awai, SUSE, 2003
- Linux Scheduler Latency - by Clark Williams, Red Hat, March 2002
- Realfeel Test of the Preemptible Kernel Patch - article in Linux Journal, 2002 by Andrew Webber
  - This is a test of the preemptible kernel feature in 2.4.19, on i386 hardware.
- Real Time and Linux, Part 3: Sub-Kernels and Benchmarks
  - article in Embedded Linux Journal, online, 2002 by Kevin Dankwardt
- [attachment:p-a03_wilshire.pdf Real Time Linux: Testing and Evaluation] - By Phil Wilshire of Lineo at the Second Real Time Linux Workshop, 2000
  - This paper discusses the different benchmarking tools used to evaluate the performance of Linux and their suitability for evaluating Real Time system Performance. It is focused on RTAI.

OLS papers

[FIXTHIS - need to scan for past papers]

- OLS 2006 BOF - Steven Rostedt, RedHat and Klaas Van Gend, MontaVista
  - See The State of RT and Common Mistakes (OLS 2006 BOF)
- OLS 2007 - Paper by Steven Rostedt - see https://ols2006.108.redhat.com/2007/Reprints/rostedt-Reprint.pdf

Darren Hart wrote:

```
I have contributed some testing results to Steven Rostedt's OLS RT Internals
paper.  That will be available to link to after the conference sometime.
```

Real Time Linux Foundation RTL Workshops

Nicholas said:

There are a number of publications related to both benchmarking and analysis of hardware related artifacts (cache,BTB,TLB,etc.) which were published at the real-time Linux Workshops.

Here is a link to the RTLF events page:

- http://www.realtimelinuxfoundation.org/events/events.html

So far, I've scanned 1999-2000 for interesting links.

Uncategorized stuff

This section has random stuff I haven't organized yet:

- http://eaglet.rain.com/rick/linux/schedstat/
  - scheduler statistics
  - maybe this can be used to analyze process wakeup latency?? Need to see what stats are kept.
- Low-latency HowTo (for audio) - http://lowlatency.linuxaudio.org/

Notes on ineffective tests

Nicholas McGuire wrote:

```
The tests noted in the LKML post on this page are very problematic,
ping - -f is not testing RT at all, it keeps the kernel in a very small active
page set thus reducing page related penalties, the while loop using dd
is also not too helpfull as it will de-facto run only in memory and cause
absolutely no disk/mass-storage related interaction (try the same with
mount -o remount,sync /  first and it will be devastating ! (limited to ext2/ext3/ufs))
```

Notes on test requirements - need to test kernel error paths

Nicholas McGuire wrote:

```
The big problem with RT tests published is that they are all looking at the good case,
they are loading the system but assuming successfull operations. The worst cases pop
up when you run in the error paths of the kernel - then a trivial application can
induce very large jitter in the system (run crashme in the background and rerun
the tests...)
```

Notes on test requirements - need for usage profile

Also lmbench can give a statistic view of things (and not even that very precisely in some case i.e. context switch measurements are flawed) so this is not of much help for descision makers which variant to use - it does not help if the average performance is good but the mobile phone or mp3 klicks at 1s intervals "deterministically" - so I guess RT benchmarks need a notion of usage-profile to be of value.

Category:

- Linux

> From: eLinux.org

RT-Preempt Tutorial

A summary tutorial of techniques for using rt-preempt and real-time with Linux.

The video of Kevin Dankwardt speaking on the use of RT-Preempt

The html slides and odp slides from that talk.

The sample programs, and the rt-preempt patch, from the summary can be found here.

The kernel used in that example

Category:

- HOWTOs

> From: eLinux.org

Soft IRQ Threads

Contents

Introduction

This page describes Soft IRQ threads, which is a mechanism to run certain interrupt bottom halves using a kernel thread with scheduling that can be prioritized in the same scheduling class as other threads and processes.

LKML discussions

Recently (July 2004), two patches have been submitted to provide Soft IRQ threads for the Linux kernel.

Time Sys patch

See http://lkml.org/lkml/2004/7/13/125

It applies against 2.6.8-mm1, with one PPC-specific reject

Korty patch

See http://lkml.org/lkml/2004/7/13/152

Rationale

This feature is important because it allows Soft IRQ processing to be scheduled at a lower priority than realtime threads. This allows for better realtime handling by the Linux kernel.

Downloads

Patch

```
- [Patch for CELF version XXXXXX is *here*]
- [Patch for 2.4.xx is *here*]
- [Patch for 2.6.xx is *here*]
```

Utility programs

None

How To Use

[Should fill this in]

Sample Results

Future Work

Here is a list of things that could be worked on for this feature:

```
(None so far)
```

Category:

- Kernel

> From: eLinux.org

Ti AM33XX PRUSSv2

The PRUSS (Programmable Real-time Unit Sub System) consists of two 32-bit 200MHz real-time cores, each with 8KB of program memory and direct access to general I/O. These cores are connected to various data memories, peripheral modules and an interrupt controller for access to the entire system-on-a-chip via a 32-bit interconnect bus.

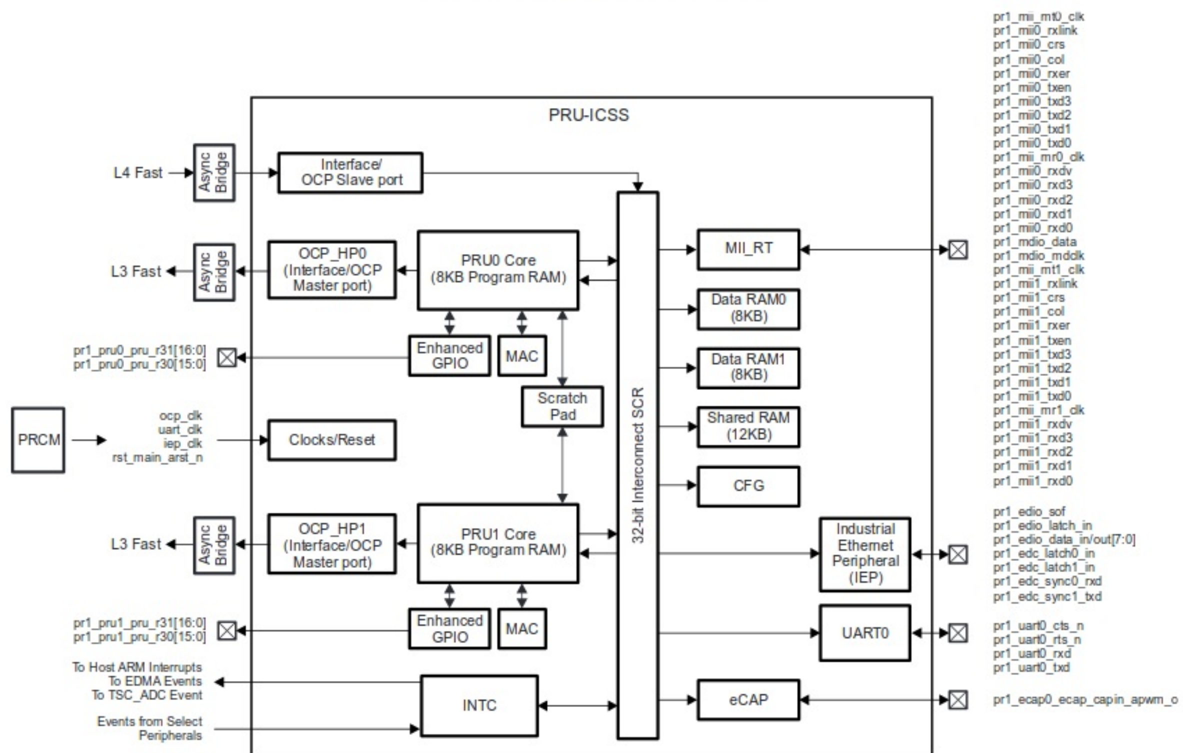
PRUs are programmed in Assembly, with most commands executing in a single cycle and with no caching or pipe-lining, allowing for 100% predictable timings. At 200MHz, most operations will take 5ns (nanoseconds) to execute, with the exception of accessing memory external to PRU.

Contents

This is a Work In Progress

Available PRU Resources

Figure 2. PRU-ICSS Integration



Click here for a full list of register mappings.

Per PRU

8KB program memory Memory used to store instructions and static data AKA Instruction Memory (IRAM). This is the memory in which PRU programs are loaded.

Enhanced GPIO (EGPIO) High-speed direct access to 16 general purpose output and 17 general purpose input pins for each PRU.

PRU0 *pr1_pru_0_pru_r30[15:0]* (PRU0 Register R30 Outputs)

pr1_pru_0_pru_r31[16:0] (PRU0 Register R31 Inputs)

PRU1 *pr1_pru_1_pru_r30[15:0]* (PRU1 Register R30 Outputs)

pr1_pru_1_pru_r31[16:0] (PRU1 Register R31 Inputs)

Hardware capture modes Serial 28-bit shift in and out.

Parallel 16-bit capture on clock.

MII standardised capture mode, used for implementing media independent Fast Ethernet (100Mbps - 25MHz 4-bit).

A 32-bit multiply and accumulate unit (MAC) Enables single-cycle integer multiplications with a 64-bit overflow (useful for decimal results).

8KB data memory Memory used to store dynamic data. Is accessed over the 32-bit bus and so not single-cycle.

One PRU may access the memory of another for passing information but it is recommend to use scratch pad or shared memory, see below.

Open Core Protocol (OCP) master port Access to the data bus that interconnects all peripherals on the SoC, including the ARM Cortex-A8, used for data transfer directly to and from the PRU in Level 3 (L3) memory space.

Shared Between PRUs

Scratch pad 3 banks of 30 32-bit registers (total 90 32-bit registers).

Single-cycle access, can be accessed from either PRU for data sharing and signalling or for individual use.

12KB data memory Accessed over the 32-but bus, not single-cycle.

Local Peripherals

Local peripherals are those present within the PRUSS and not those belonging to the entire SoC. Peripherals are accessed from PRUs over the Switched Central Resource (SCR) 32-bit bus within the PRUSS.

Attached to the SCR bus is also an OCP slave, enabling OCP masters from outside of the PRUSS to access these local peripherals in Level 4 (L4) memory space.

Enhanced Capture Model (eCAP)

Industrial Ethernet Peripheral (IEP)

Universal Asynchronous Receiver/Transmitter (UART0) Used to perform serial data transmission to the TL16C550 industry standard.

16-bit FIFO receive and transmit buffers + per byte error status.

Can generate Interrupt requests for the PRUSS Interrupt Controller.

Can generate DMA requests for the EDMA SoC DMA controller.

Maximum transmission speed of 192MHz (192Mbps - 24MB/s).

Communication

Communication between various elements of the PRUSS or the wider SoC may take place either directly, over a bus, via interrupts or via DMA.

The following lists will expose all possible communication approaches for each likely scenario.

For communication via interrupts, please first read the section on the PRUSSv2 Interrupt Controller.

Click here for a full list of PRUSS Interrupts.

The current example PRU loader uses UIO, but this ideally should be replaced with remoteproc rather than poking at the registers from userspace. In the mean time, according to this discussion: we can use the included script and load the uio_pruss userspace driver.

PRU to Host (PRU to ARM Cortex-A8)

Include the uio_pruss kernel driver by using **modprobe uio_pruss** or the steps outlined above, if that does not work. Then in a project include the header files for the am335x_pru_package.

```
#define PRU_NUM0      0
// Driver header file
#include <prussdrv.h>
#include <pruss_intc_mapping.h>
```

/* Then, initialize the interrupt controller data */

```
tpruss_intc_initdata pruss_intc_initdata = PRUSS_INTC_INITDATA;
```

/* Initialize the PRU */

```
prussdrv_init ();
```

/* Get the interrupt initialized */

```
prussdrv_pruintc_init(&pruss_intc_initdata)
```

/* Execute example on PRU0 where first argument is the PRU# and second is the assembly to execute*/

```
prussdrv_exec_program (PRU_NUM0, "./example.bin");
```

/* Wait until PRU0 sends the interrupt*/

```
prussdrv_pru_wait_event (PRU_EVTOUT_0);
```

/* Clear the interrupt*/

```
prussdrv_pru_clear_event (PRU0_ARM_INTERRUPT);
```

The PRU (in this case 0) will have the following in the example.bin file to trigger the interrupt:

```
#define PRU0_ARM_INTERRUPT      19
MOV       r31.b0, PRU0_ARM_INTERRUPT+16
```

Register 31 allows for control of the INTC for the PRU.

Host to PRU (ARM Cortex-A8 to PRU)

Interrupts

Each PRU has access to host interrupt channels Host-0 and Host-1 through register R31 bit 30 and bit 31 respectively. By probing these registers, a PRU can determine if an interrupt is currently present on each host channel.

To configure

PRU to external peripherals

External peripherals to PRU

PRU to internal peripherals

Internal peripherals to PRU

Loading a PRU Program

Beaglebone PRU connections and modes

PRU #	R30(output) bit	Pinmux Mode	R31(input) bit	Pinmux Mode	BB Header	BB Pin Name	ZCZ BallName
0	0	Mode_5	0	Mode_6	P9_31	SPI1_SCLK	mcasp0_aclkx
0	1	Mode_5	1	Mode_6	P9_29	SPI1_D0	mcasp0_fsx
0	2	Mode_5	2	Mode_6	P9_30	SPI1_D1	mcasp0_axr0
0	3	Mode_5	3	Mode_6	P9_28	SPI1_CS0	mcasp0_ahclkr
0	4	Mode_5	4	Mode_6	P9_42	(*note1)	mcasp0_aclkr
0	5	Mode_5	5	Mode_6	P9_27	GPIO3_19	mcasp0_fsr
0	6	Mode_5	6	Mode_6	P9_41	(*note2)	mcasp0_axr1
0	7	Mode_5	7	Mode_6	P9_25	GPIO3_21	mcasp0_ahclkx
0	14	Mode_6	N/A		P8_12	GPIO1_12	gpmc_ad12
0	15	Mode_6	N/A		P8_11	GPIO1_13	gpmc_ad13
0	N/A		14	Mode_6	P8_16	GPIO1_14	gpmc_ad14
0	N/A		15	Mode_6	P8_15	GPIO1_15	gpmc_ad15
0	N/A		16	Mode_6	P9_24	UART1_TXD	uart1_txd
1	0	Mode_5	0	Mode_6	P8_45	GPIO2_6	lcd_data0
1	1	Mode_5	1	Mode_6	P8_46	GPIO2_7	lcd_data1
1	2	Mode_5	2	Mode_6	P8_43	GPIO2_8	lcd_data2
1	3	Mode_5	3	Mode_6	P8_44	GPIO2_9	lcd_data3
1	4	Mode_5	4	Mode_6	P8_41	GPIO2_10	lcd_data4
1	5	Mode_5	5	Mode_6	P8_42	GPIO2_11	lcd_data5
1	6	Mode_5	6	Mode_6	P8_39	GPIO2_12	lcd_data6
1	7	Mode_5	7	Mode_6	P8_40	GPIO2_13	lcd_data7
1	8	Mode_5	8	Mode_6	P8_27	GPIO2_22	lcd_vsync
1	9	Mode_5	9	Mode_6	P8_29	GPIO2_23	lcd_hsync
1	10	Mode_5	10	Mode_6	P8_28	GPIO2_24	lcd_pclk
1	11	Mode_5	11	Mode_6	P8_30	GPIO2_25	lcd_ac_bias_en
1	12	Mode_5	12	Mode_6	P8_21	GPIO1_30	gpmc_csn1
1	13	Mode_5	13	Mode_6	P8_20	GPIO1_31	gpmc_csn2
1	N/A		16	Mode_6	P9_26	UART1_RXD	uart1_rxd

```
*Note1: The PRU0 Registers{30,31} Bit 4 (GPIO3_18) is routed to P9_42-GPIO0_7 pin.  You MUST set GPIO0_7 to input mod
e in pinmuxing.

*Note2: The PRU0 Registers{30,31} Bit 6 (GPIO3_20) is routed to P9_41-GPIO0_20(CLKOUT2). You must set GPIO0_20 to inp
ut mode in pinmuxing.
```

Assembly

The complete list of PRU assembly instructions can be found at TI.

Four instruction classes

- Arithmetic
- Logical
- Flow Control
- Register Load/Store

Instruction Syntax

- Mnemonic, followed by comma separated parameter list
- Parameters can be a register, label, immediate value, or constant table entry
- Example
    - SUB r3, r4, 10
    - Subtracts immediate value 10 (decimal) from the value in r4 and then places the result in r3 (or r3 = r4 - 10)

C Compiler

TI

- TI C Compiler download
- TI Code Composer Studio
- CCS PRU compiler release discussion on Element14 site

GCC

- GCC C wiki
- GCC C compiler source
- GCC C compiler announcement

Forth Compiler

- Forth for PRU

Resources

- PRU FAQ
- AM335x PRU support package on Github
- AM335x PRU-ICSS Reference Guide
- The documentation on the subsystem is here. TI does not support this subsystem and all questions/inquires/problems should be directed to the community.
- Example for the first PRU version (Ti AM18XX and other DSP) http://www.ti.com/tool/sprc940
- Element14 write-up on using PRU
- PRU assembly syntax highlighting for TextMate, Sublime Text, etc.
- A userspace debugging utility with credit to PRU_EVTOUT_2 from the #beagle IRC channel.
- ncurses based debugger work has started at here.
- A classic CLI debugger is available on SourceForge called prudebug.
- For using the Open Core Protocol to access external memory from the PRU.
- Jason Kridner's presentation at Pumping Station: One - video slides

Examples

- BeagleBone_6502_RemoteProc_cape
- BeagleBone Nixie Cape PRU App
- PRU Soft UART Driver
- PRU Projects
- SPI ADC

Categories:

- ECE497
- BeagleBone
- PRU

> From: eLinux.org

Variable Scheduling Timeouts

Rationale

The Linux kernel relies on periodic timer interrupts to allow the kernel:

- to keep the current time and date up to date,
- to do accounting,
- to check if a different task should be dispatched, and
- to see if software timer service routines should be called.

Even when the processor is otherwise idle, these interrupts occur and cause the processor to be brought out of a power savings mode even though there may not be anything for it to do other than update the time and go back to sleep. The Variable Scheduling Timeouts (VST) feature modifies the kernel to avoid unnecessary timer interrupts, allowing the processor to stay in a power saving mode for longer periods.

Specification

A kernel SHOULD disable periodic timer interrupt while CPU is idle and there is no scheduled timer. The length of the time interval during which periodic timer interrupt is disabled SHOULD not be more than a value which SHOULD be configurable.

Notes

More information on VST is available at the community site http://sourceforge.net/projects/high-res-timers/

Category:

- ObsoleteInfo

> From: eLinux.org

Boot Time

Contents

Introduction

Boot Time includes topics such as measurement, analysis, human factors, initialization techniques, and reduction techniques. The time that a product takes to boot directly impacts the first perception an end user has of the product. Regardless of how attractive or well designed a consumer electronic device is, the time required to move the device from off to an interactive, usable state is critical to obtaining a positive end user experience. Turning on a device is Use Case #1.

Booting up a device involves numerous steps and sequences of events. In order to use consistent terminology, the Bootup Time Working Group of the CE Linux Forum came up with a list of terms and their widely accepted definitions for this functionality area. See the following page for these terms:

- Boot-up Time Definition Of Terms

Technology/Project Pages

The following are individual pages with information about various technologies relevant to improving Boot Time for Linux. Some of these describe local patches available on this site. Others point to projects or patches maintained elsewhere.

Measuring Boot-up Time

- Printk Times - simple system for showing timing information for each printk.
- Kernel Function Trace - system for reporting function timings in the kernel.
- Linux Trace Toolkit - system for reporting timing data for certain kernel and process events.
- Oprofile - system-wide profiler for Linux.
- Bootchart - a tool for performance analysis and visualization of the Linux boot process. Resource utilization and process information are collected during the user-space portion of the boot process and are later rendered in a PNG, SVG or EPS encoded chart.
- Bootprobe
  - a set of System Tap scripts for analyzing system bootup.
- and, let us not forget: "cat /proc/uptime"
- grabserial
  - a nice utility from Tim Bird to log and timestamp console output
- process trace
  - a simple patch from Tim Bird to log exec, fork and exit system calls.
- ptx_ts - Pengutronix' TimeStamper: A small filter prepending timestamps to STDOUT; a bit similar to grabserial but not limited to serial ports
- Initcall Debug - a kernel command line option to show time taken for initcalls.
- See also: Kernel Instrumentation which lists some known kernel instrumentation tools. These are of interest for measuring kernel startup time.

Technologies and Techniques for Reducing Boot Time

Bootloader speedups

- Kernel XIP - Allow kernel to be executed in-place in ROM or FLASH.
- DMA Copy Of Kernel On Startup
  - Copy kernel from Flash to RAM using DMA
- Uncompressed kernel - An uncompressed kernel might boot faster
- Fast Kernel Decompression

Kernel speedups

- Disable Console - Avoid overhead of console output during system startup.
- Disable bug and printk - Avoid the overhead of bug and printk. Disadvantage is that you lose a lot of info.
- RTC No Sync - Avoid delay to synchronize system time with RTC clock edge on startup.
- Short IDE Delays - Reduce duration of IDE startup delays (this is effective but possibly dangerous).
- Hardcode kernel module info - Reduce the overhead of loading a module, by hardcoding some information used for loading the relocation information
- IDE No Probe - Force kernel to observe the ide\=noprobe option.
- Preset LPJ - Allow the use of a preset loops_per_jiffy value.
- Asynchronous function calls - Allow probing or other functions to proceed in parallel, to overlap time-consuming boot-up activities.
  - Threaded Device Probing - Allow drivers to probe devices in parallel. (not mainlined, now deprecated?)
- Reordering of driver initialization
  - Allow driver bus probing to start as soon as possible.
- Deferred Initcalls - defer non-essential module initialization routines to after primary boot
- NAND ECC improvement - The pre 2.6.28 nand_ecc.c has room for improvement. You can find an improved version in the mtd git at http://git.infradead.org/mtd-2.6.git?a=blob_plain;f=drivers/mtd/nand/nand_ecc.c;hb=HEAD. Documentation for this is in http://git.infradead.org/mtd-2.6.git?a=blob_plain;f=Documentation/mtd/nand_ecc.txt;hb=HEAD. This is only interesting if your system uses software ECC correction.
- Check what kernel memory allocator you use. Slob or slub might be better than slab (which is the default in older kernels)

- If your system does not need it, you can remove SYSFS and even PROCFS from the kernel. In one test removing sysfs saved 20 ms.
- Carefully investigate all kernel configuration options on whether they are applicable or not. Even if you select an option that is not used in the end, it contributes to the kernel size and therefore to the kernel load time (assuming you are not doing kernel XIP). Often this will require some trial and measure! E.g. selecting CONFIG_CC_OPTIMIZE_FOR_SIZE (found under general setup) gave in one case a boot improvement of 20 ms. Not dramatic, but when reducing boot time every penny counts!
- Moving to a different compiler version might lead to shorter and/or faster code. Most often newer compilers produce better code. You might also want to play with compiler options to see what works best.
- If you use initramfs in your kernel and a compressed kernel it is better to have an uncompressed initramfs image. This is to avoid having to uncompress data twice. A patch for this has been submitted to LKML. See http://lkml.org/lkml/2008/11/22/112

File System issues

Different file systems have different initialization (mounting) times, for the same data sets. This is a function of whether meta-data must be read from storage into RAM or not, and what algorithms are used during the mount procedure.

- Filesystem Information - has information about boot-up times of various file systems
- File Systems - has information on various file systems that are interesting for embedded systems. Also includes some improvement suggestions.
- Avoid Initramfs - explains on why initramfs should be avoided if you want to minimize boot time
- Split partitions. If mounting a file system takes long, you can consider splitting that filesystem in two parts, one with the info that is needed during or immediately after boot, and one which can be mounted later on.
- Ramdisks demasked - explains why using a ram disk generally results in a longer boot time, not a shorter one.

User-space and application speedups

- Optimize RC Scripts - Reduce overhead of running RC scripts
- Parallel RC Scripts - Run RC scripts in parallel instead of sequentially
- Application XIP - Allow programs and libraries to be executed in-place in ROM or FLASH
- Pre Linking - Avoid cost of runtime linking on first program load
- Statically link applications. This avoids the costs of runtime linking. Useful if you have only a few applications. In that case it could also reduce the size of your image as no dynamic libraries are needed
- GNU_HASH: ~ 50% speed improvement in dynamic linking
  - See http://sourceware.org/ml/binutils/2006-06/msg00418.html
- Application Init Optimizations
  - Improvements in program load and init time via:
  - use of mmap vs. read
  - control over page mapping characteristics.
- Include modules in kernel image
  - Avoid extra overhead of module loading by adding the modules to the kernel image
- Speed up module loading - Use Alessio Igor Bogani's kernel patches to improve module loading time by "Speed up the symbols' resolution process" (Patch 1, Patch 2, Patch 3, Patch 4, Patch 5).
- Avoid udev, it takes quite some time to populate the /dev directory. In an embedded system it is often known what devices are present and in any case you know what drivers are available, so you know what device entries might be needed in /dev. These should be created statically, not dynamically. mknod is your friend, udev is your enemy.
- If you still like udev and also like fast boot-up's, you might go this way: start your system with udev enabled and make kind of a backup of the created device nodes. Now, modify your init script like this: instead running udev, copy the device nodes that you just made a backup of into the device tree. Now, install the hotplug-daemon like you always do. That trick avoids the device node creation at startup but stills lets your system create device nodes later on.
- If your device has a network connection, preferably use static IP addresses. Getting an address from a DHCP server takes additional time and has extra overhead associated with it.
- Moving to a different compiler version might lead to shorter and/or faster code. Most often newer compilers produce

better code. You might also want to play with compiler options to see what works best.
- If possible move from glibc to uClibc. This leads to smaller executables and hence to faster load times.
- library optimiser tool: http://libraryopt.sourceforge.net/

  This will allow you to create an optimised library. As unneeded functions are removed this should lead to a performance gain. Normally there will be library pages which contain unused code (adjacent to code that is used). After optimizing the library this does not occur any more, so less pages are needed and hence less page loads, so some time can be saved.

- Function reordering: http://www.celinux.org/elc08_presentations/DDLink%20FunctionReorder%2008%2004.pdf

  This is a technique to rearrange the functions within an executable so they appear in the order they are needed. This improves the load time of the application as all initialization code is grouped into a set of pages, instead of being scattered over a number of pages.

Suspend related improvements

Another approach to improve boot time is to use a suspend related mechanism. Two approaches are known.

- Using the standard hibernate/resume approach. This is what has been demonstrated by Chan Ju, Park, from Samsung. See sheet 23 and onwards from this PPT and section 2.7 of this paper.

  Issue with this approach is that flash write is much slower than flash read, so the actual creation of the hibernate image might take quite a while.

- Implementing snapshot boot. This is done by Hiroki Kaminaga from Sony and is described at snapshot boot for ARM and http://elinux.org/upload/3/37/Snapshot-boot-final.pdf This is similar to hibernate and resume, but the hibernate file is retained and used upon every boot. Disadvantage is that no writable partitions should be mounted at the time of making the snapshot. Otherwise inconsistencies will occur if a partition is modified, while applications in the hibernate file might have information in the snapshot related to the unmodified partition.

Miscellaneous topics

About Compression discusses the effects of compression on boot time. This can affect both the kernel boot time as well as user-space startup.

Uninvestigated speedups

This section is a holding pen for ideas for improvement that are not implemented yet but that could result in a boot time gain. Please leave a note here if you are working on one of these items to avoid duplicate work.

- **Prepopulated buffer cache** - As initramfs performs an additional copy of the data the idea is to have a prepopulated buffer cache. A simplistic scenario would allow dumping the buffer cache when the booting is completed and the user applications have initialised. This data then could be used in a subsequent boot to initialize the buffer cache (of course without copying). A possible approach would be to have those data to reside into the kernel image and use them directly. Alternately they could be loaded separately. Unfortunately my knowledge of the internals in this section is not yet good enough to do a trial implementation. Caveats:
  - is it possible to have the buffer cache split into two different parts, one which is statically allocated, one which is dynamically allocated?
  - the pages in the prepopulated buffer cache probably cannot be discarded, so they should be pinned
  - apart from the buffer cache data itself also some other variables might need restoring
  - a similar approach could also be used for the cached file data.
- **Dedicated fs** - currently a lot of abstraction is done in fs to make a nice abstraction allowing easy addition of new filesystems and creating a unified view of those filesystem. While this is pretty neat, the abstraction layers also introduce some overhead. A solution could be to create a dedicated fs system, which supports only one (or maybe 2) filesystems, and eliminates the abstraction overhead. This will give some benefit, but the chance of getting this into the mainline is zero.

Articles and Presentations

- Embedded Linux boot time reduction workshop materials
    - By Free Electrons
    - Presentation on boot time reduction techniques - Practical labs on Atmel SAMA5 hardware.
- "Boot Time Optimizations" - (Slides | Video)
    - Alexandre Belloni has presented at ELC Europe on Nov 6, 2012
    - Main link at Free-Electrons
- "The Right Approach to Boot Time Reduction" - (Slides | YouTube Video)
    - Andrew Murray has presented at ELC Europe on October 28, 2010 (Free Electrons video here)
    - This included a \< 1 second QT cold Linux boot case study for an SH7724 with some additional information about 'function re-ordering' in user-space
    - Similar slides with \< 1 second case study for OMAP3530EVM can be found here
- "One Second Linux Boot Demonstration (new version)" (Youtube video by MontaVista)
- "Tools and Techniques for Reducing Bootup Time" (PPT | ODP | PDF | video)
    - Tim Bird has presented at ELC Europe, on November 7, 2008, his latest collection of tips and tricks for reducing bootup time
    - Tims Fastboot Tools has online materials in support of this presentation
- Christopher Hallinan has done a presentation at the MontaVista Vision conference 2008 on the topic of reducing boot time. Slides available here
- Optimizing Linker Load Times
    - (introducing various kinds of bootuptime reduction, prelinking, etc.)
- Benchmarking boot latency on x86
    - By Gilad Ben-Yossef, July 2008
    - A tutorial on using TSC register and the kernel PRINTK_TIMES feature to measure x86 system boot time, including BIOS, bootloader, kernel and time to first user program.
- Fast Booting of Embedded Linux
    - By HoJoon Park, Electrons and Telecommunications Research Institute (ETRI), Korea, Presented at the CELF 3rd Korean Technical Jamboree, July 2008
    - Explains several different reduction techniques used for different phases of bootup time
- Tim Bird's (Sony) survey of boot-up time reduction techniques:
    - Methods to Improve Boot-up Time in Linux
        - Paper prepared for 2004 Ottawa Linux Symposium
    - http://elinux.org/images//8/83/Pdf.gif Reducing Startup Time in Embedded Linux Systems http://elinux.org/images/d/da/Info_circle.png
        - December 2003 Presentation describing some existing boot-up time reduction techniques and strategies.
- Parallelizing Linux Boot on CE Devices
    - PDF of Presentation
    - Video of Presentation
- Parallelize Applications for Faster Linux Boot
    - Authored by M. Tim Jones for IBM Developer Works
    - This article shows you options to increase the speed with which Linux boots, including two options for parallelizing the initialization process. It also shows you how to visualize graphically the performance of the boot process.
- Android Boot Time Optimization
    - Authored by Kan-Ru Chen, 0xlab
    - This presentation covers Android boot time measurement and analysis, the proposed reduction approaches, hibernation-based technologies, and potential Android user-space optimizations.
- Texas Instruments Embedded Processors Wiki provides the procedure to optimize Linux/Android boot time:
    - Optimize Linux Boot Time
    - Android Boot Time Optimization
- Implement Checkpointing for Android
    - Authored by Kito Cheng and Jim Huang, 0xlab
    - Reasons to Implement Checkpointing for Android

- Resume to stored state for faster Android boot time
- Better product field trial experience due to regular checkpointing

Case Studies

- 300 milliseconds from boot loader to shell on ARM with NAND
- Samsung proof-of-acceptability study for digital still camera: see Boot Up Time Reduction PPT and the paper describing this.
- Boot Linux from Processor Reset into user space in less than 1 Second
    - In this white paper, Robin Getz describes the techniques used to fast-boot a blackfin development board.
- Booting Linux dm365 Network Camera in 3.2 seconds
- Boot of kernel and shell in 0.5 sec (not including u-boot and decompression)
- Warp2, Lineo Solutions, 2008. 2.97 sec boot, ARM11, 400MHz

Additional Projects/Mailing Lists/Resources

Replacements for SysV 'init'

The traditional method of starting a Linux system is to use /sbin/init, which processes the file /etc/inittab. This is an init program which processes a series of actions for different run-levels and system events (key-combinations and power events).

See the init(8) man page and the the inittab(5) man page.

busybox init

An 'init' applet is often included in BusyBox

There used to be (as of 2000) some slight differences in the supported features of the 'inittab' file between busybox init and full-blown init. However, I don't know (as of 2010) if that's still the case. (See http://spblinux.de/2.0/doc/init.html for some details)

Denys Vlasenko, one of the maintainers of busybox has suggested a replacement for traditional init for that tool called runsv. See http://busybox.net/~vda/init_vs_runsv.html

upstart

upstart is the name of a newer Linux desktop systems that provides the program /sbin/init, but with different operational semantics.

- Home page: http://upstart.ubuntu.com/
- Wikipedia page: http://en.wikipedia.org/wiki/Upstart

Android init

Android 'init' is a custom program for booting the Android system.

See Android 'init'

systemd

systemd is a new project (as of May 2010) for starting daemons and services on a Linux desktop system

See http://0pointer.de/blog/projects/systemd.html

Kexec

- Kexec is a system which allows a system to be **rebooted** without going through BIOS. That is, a Linux kernel can directly boot into another Linux kernel, without going through firmware. See the white paper at: kexec.pdf
  - 2004 Kernel Summit presentation: fastboot.pdf
  - here's another Kexec white paper:Reboot Fast

Splash Screen projects

- Splashy - Technology to put up a splash screen early in the boot sequence. This is user-space code.
  - This seems to be the most current splash screen technology, for major distributions. A framebuffer driver for the kernel is required.
- Gentoo Splashscreen - newer technology to put a splash screen early in the boot sequence
  - See the HOWTO at: HOWTO FBSplash
- PSplash - PSplash is a userspace graphical boot splash screen for mainly embedded Linux devices supporting a 16bpp or 32bpp framebuffer.
- bootsplash.org - put up a splash screen early in boot sequence
  - This project requires kernel patches
  - This project is now abandoned, and work is being done on Splashy.

Others

- FSMLabs Fastboot - press release by FSMLabs about fast booting of their product. Is any of this published?

- snapshot boot - a technology uses software resume to boot up the system quickly.

Apparently obsolete or abandoned material

- ⚠ *in progress* - Boot-up Time Reduction Howto
  - this is a project to catalog existing boot-up time reduction techniques.
  - Was originally intended to be the authoritative source for bootup time reduction information.
  - No one maintains it any more (as of Aug, 2008)
- ⚠*no content yet* - Boot-up Time Delay Taxonomy
  - list of delays categorized by boot phase, type and magnitude
  - Was to be a survey of common bootup delays found in embedded devices.
  - Was never really written.

???

- Bootup Time Spec
- Bootup Time Things To Investigate
- Bootup Time Working Group
- Bootup Time Task List
- Bootup Time Howto Task List
- Fast Booting Translation

Companies, individuals or projects working on fast booting

- Intel - Arjan van de Ven - see http://lwn.net/Articles/299483/
- Tripeaks - see http://www.linuxdevices.com/news/NS8282586707.html
- Lineo Solutions - see http://www.linuxdevices.com/news/NS5185504436.html
- Monta Vista - see http://www.linuxdevices.com/news/NS2560585344.html
- fastboot git tree - see http://lwn.net/Articles/299591/
- MPC Data SwiftBoot services - http://www.swiftboot.com/

- Free Electrons - http://free-electrons.com/services/boot-time/

Boot time check list

From an August 2009 discussion about boot time on ARM devices, several hints and advice regarding boot time optimization are available. While it may repeat a lot of above, below is a check list extracted from this discussion:

- Is CPU's clock switched to maximum? If the kernel, bootloader or hardware is in charge of setting CPU power and speed scaling, then you should check that it boots with the CPU set at maximum speed instead of slowest.
- Is your hardware (register) timing configuration of your SoC's memory interfaces (e.g. RAM and NOR/NAND timing) optimized? A lot of vendors ship their hardware with "well, it works, optimize later" settings. What you want is "as fast as possible, but sill stable and reliable" configuration. This might need some hardware knowledge and has to be customized to the individual memory devices used.
- Does your boot loader uses I- and D-Cache? E.g. U-Boot doesn't enable D-Cache by default on ARM devices, as it needs customized MMU tables to do so.
- Does kernel copy from permanent storage (e.g. NOR or NAND) to RAM use optimized functions? E.g. DMA, or on ARM at least load/store multiple commands (ldm/stm)?
- If you use U-Boot's uImage, set "verify=no" in U-Boot to avoid checksum verification.
- Optimize size of your kernel.
    - You might even try some of the embedded system kernel config options that, for example, eliminate all the printk strings, reduce data structures, or eliminate unneeded functionality.
- How often is kernel (image) data copied? First by boot loader from storage to RAM, then by kernel's uncompressor to it's final destination? Once more? If you use compressed kernel and NOR flash, consider running the uncompressor XIP in NOR flash.
- If you use compressed kernel, check compression algorithm. zlib is slow on decompression, and lzo is much faster. So if you implement lzo compression, you'll probably speed things up a little as well (check LKML for this). Having no compression at all may also be a good thing to try (see next topic).
- Check to use uncompressed kernel (depends on your system configuration). Using an uncompressed kernel on a flash-based system may improve boot time. The reason is that compressed kernels are faster only when the throughput to the persistent storage is lower than the decompression throughput, and on typical embedded systems with DMA the throughput to memory outperforms the CPU-based decompression. Of course it depends on a lot of stuff like performance of flash controller, kernel storage filesystem performance, DMA controller performance, cache architecture etc. So it's individual per-system. Example: With using an uncompressed kernel (~2.8MB) uncompressing (running the uncompressor XIP in NOR flash) took ~0.5s longer than copying 2.8MB from flash to RAM.
- Enable precalculated loops-per-jiffy
- Enable kernel quiet option
- If you use UBI: UBI is rather slow in attaching MTD devices. Everything is explained at MTD's UBI scalability and UBI fs scalability sections. There is not very much you can do to speed it up but implement UBI2. UBIFS would stay intact. There were discussions about this and it does not seem to be impossibly difficult to do UBI2 (few ideas).
    - In a follow-up e-mail, Sascha Hauer wrote:

> "What's interesting about this is that the kernel NAND driver is much slower
>
> than the one in U-Boot. Looking at it it turned out that the kernel driver uses interrupts to wait for the controller to get ready. Switching this to polling nearly doubles the NAND performance. UBI mounts much faster now and this cuts off another few seconds from the boot process :) "

- Use static device nodes during boot, and later setup busybox mdev for hotplug.

- If you have network enabled, there might be some very long timeouts in the network code paths, which appear to be used whether you specify a static address or not. See the definitions of CONF_PRE_OPEN and CON_POST_OPEN in *net/ipv4/ipconfig.c*. Check ipdelay configuration patch.
- Parallelize boot process.
- Disable the option "Set system time from RTC on startup and resume", you can use the command hwclock -s at the of the init instead of slowing down the kernel.

Categories:

- Boot Time
- Bootloader
- CE Linux Working Groups

> From: eLinux.org

About Compression

Contents

About Compression

Introduction

This page discussed compression in relation to boot and load time. First we explain the effects of compression on the system in general. Next we discuss where compression may be applied. After that we will discuss the alternatives and go into detail whether they are worthwhile or not.

Effects of compression (general discussion)

Space impact

Generally compression will cause that less space is needed. Typically a piece of data that is compressed becomes smaller. Note though that this is not always the case. If you compress something which is already compressed there is probably nothing to gain, and in most cases you'll lose a little bit, because the compressor itself adds info for decompression. Compressing audio or video files (like mp3, DivX or jpeg files) also will yield little space reduction as these data files are internally already highly compressed.

The amount of compression also depends on the compression algorithm and the options that are applied.

Performance impact

Of course compression also has a performance impact. At first sight it may seem that the impact is negative, as additional time is needed to compress/decompress the data. However experiments may yield otherwise.

Consider the following situation. You have a 2 MB linux kernel. Compression will reduce that kernel to 1 MB. Now suppose you store the kernel in flash which has a read speed of 20 MB/s. In case of an uncompressed kernel, 100 ms will be needed to read the kernel from flash to RAM. However if the kernel is compressed, the actual I/O activity will only take 50 ms. So if decompressing 1MB takes less than 50 ms (decompression speed > 40 MB/s) you gain some time.

So whether or not compression affects your performance positive or negative depends on factors like read speed and decompression speed.

Of course it is not as simple as this. If reading is done through DMA and your system is CPU constrained compression will cause performance degradation. This is because reading using DMA is almost free as far as the CPU is concerned, and the decompression additionally loads the CPU (which was already the bottleneck).

Where compression may be applied

There are a number of places where compression can be applied in Linux:

- If you use an initramfs in the kernel by default this initramfs is compressed.
- The kernel itself often is also compressed. (hint: if your kernel name is vmlinuz or bzImage it is most likely a compressed kernel, if your kernel name is vmlinux it is probably not compressed; if you want to make sure: **file** is your friend here.
- The filesystem itself can also apply compression. This is often the case with flash file systems. Notably SquashFS, CRAMFS and JFFS2 use filesystem compression.
- Compression of executables, application data and the like. This is outside the scope of this page and not covered.

Initramfs compression

Initramfs in most cases is compressed. If you specify an initramfs file system while building the kernel, the initramfs image will be compressed and embedded into the kernel image. It is not required to compress the image, but there is no CONFIG option to disable this. The build scripts always perform the compression, and only if you tweak the script you can avoid the compression.

Another way to specify an initramfs is through the initrd= boot line parameter. In that case it is totally up to you whether or not you use a compressed or uncompressed cpio archive.

Now is initramfs compression useful? Well the first answer depends on load time versus decompress time as explained before. However, if you are using a compressed kernel and an embedded initramfs image, compression is useless as you compress the data twice. First as initramfs image and later a second time when the kernel is compressed. This is not giving additional benefit and only wastes CPU time.

Despite the arguments above though, there is one case where a compressed initramfs image in a compressed kernel is meaningful. That is if you are low on RAM. A compressed initramfs requires less RAM so it might well be that a compressed image fits whereas an uncompressed image does not fit. Note though that this only applies to RAM usage during boot. When the bootstrapping is completed and the kernel is started the space taken up by the initramfs image is released.

Kernel compression

For kernel compression the general arguments hold. However note that kernel decompression is done by the boot loader. This means that factors like read speed and decompression performance depend on the implementation of the boot loader and not on the implementation in the kernel. As these normally differ decompression speed here might differ from the decompression done in the kernel for initramfs.

Filesystem compression

The underlying filesystem can also provide compression. So you could in theory have a compressed initramfs in a compressed kernel which reside in a compressed filesystem. Not desirable at all. If your filesystem uses compression you probably do not want to compress kernel or initramfs. Then again this is probably not likely. Most boot loaders seem to load the kernel directly from flash or from an uncompressed filesystem.

Concluding remarks

- Take care to avoid nested compression
- When deciding whether to apply compression or not, make sure to benchmark both alternatives

Category:

- System Size

> From: eLinux.org

Application Init Optimizations

Contents

Description

This page describes optimizations to a large application and to the kernel, to shorten the time required to load and execute an application.

Two main techniques are described here: 1) use of mmap vs. read and 2) control over page mapping characteristics. These techniques are discussed below.

Rationale

Kernel bootup time is drastically improved with recent efforts including CELF activities. As a next step, application bootup time should be considered to cut down the system total bootup time. The techniques described here are applicable to a large number of embedded systems, which consist of large, single-application programs.

Application Tuning

Using mmap() instead of read() for initial application data load

An application may load a large amount of data when it is first initialized. This can result in a long delay as the file data is read into memory. It is possible to avoid the initial cost of this read, by using mmap() instead of read().

Instead of loading all of the data into memory with the read system call, the file can be mapped into memory with the mmap system call. Once the data file is mapped, individual pages will be demand loaded during execution, when the application reads them. Depending on the initial working set size of the data in the file, this can result in significant time savings. (For example, if an application only initially uses 50% of the data from the file, then only 50% of the data will be read into

memory from persistent storage. There is extra overhead due to the cost of page-faults incurred in loading the pages on demand. However, this page fault overhead is offset by the savings in the number of page reads (compared to the read() case).

Customizing file cache control in the Kernel

To further improve this method, the kernel can be modified to reduce page copying and page faults.

Eliminating redundant page copies

When pages are demand loaded to a memory-mapped file, the pages are kept in memory as part of the kernel “file cache” and mapped into the requesting process's address space. If the page is accessed via a write operation, then the page in the file system cache is copied to a newly allocated memory page. (This is referred to as "copy-on-write"). The copied page can be then be freely modified by the process which maps it.

Suppose, however, that a file is mapped or accessed by only one process. Then, copying the page is redundant. In this case, we can convert the page in the file cache to a private page immediately. By utilizing this assumption (only one user for the page), the cost of the copy can be eliminated. This has the side benefit of reducing memory consumption as well.

Reducing page faults

In some cases, an individual page in the process address space is accessed first with a read operation, then with a write operation. This results in two page faults for the same page (one to load the page and move it "through" the file cache, and the other to get a local copy of the page.) By eliminating the page copy, and making the page private on the first access (whether read or write), the second page fault can be reduced.

Controlling API

The current system is experimental, in the way it manages the files affected by this caching/virtual memory customization. It would be better to control this mechanism per file or virtual memory area. The fcntl system call or mmap system call are candidates where this control could be introduced.

Resources

Projects

None.

Specifications

Boot Time

Downloads

Patch

Sorry but there is no available patch at this time.

Sample Results

Case Study 1

Hardware'

- Towa Meccs TMM1000
- SH3(7709) 133MHz
- 32MB RAM
- 64MB CF memory

Software Kernel

- 2.4.27 kernel.

Target application

- intent
- Loading data size: 8MB

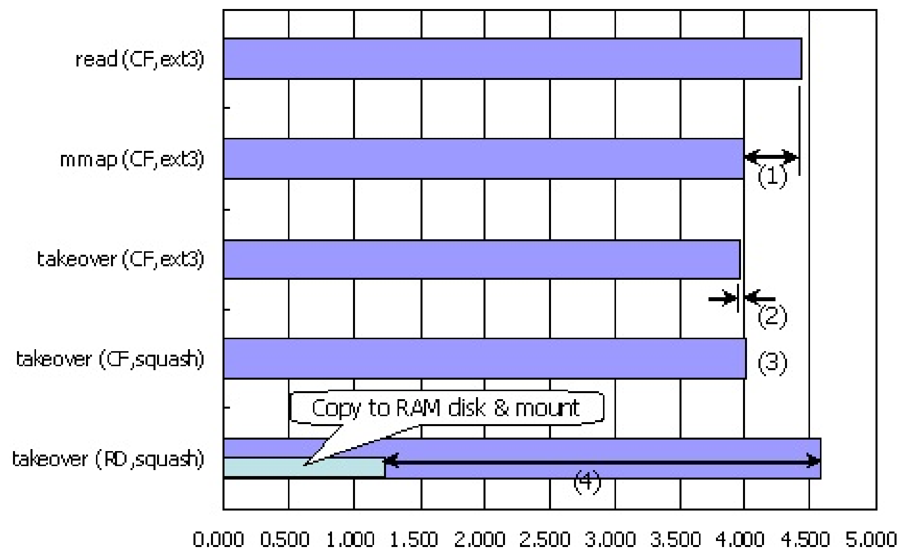
Methods explanation

1. read(CF/ext3) The data file is loaded using read system call from a ext3 file system on a CompactFlash memory.
2. mmap(CF/ext3) The data file is mapped to the process virtual space using mmap system call.
3. takeover(CF/ext3) The data file is mapped and the page in the file system cache (which is created during page fault handling) is converted to private page immediately.
4. takeover(CF/squash) Same as No.3 except using the SquashFS file system.
5. takeover(RD/squash) Same as No.3 except the file system is on read from a RAM Disk instead of Compact Flash.

Results

No.	Method	Media	FS	Ave.	1st	2nd	3rd	Diff.
1	read	CF	ext3	**4.420**	4.418	4.420	4.421	-
2	mmap	CF	ext3	**3.995**	3.995	3.995	3.996	-0.424
3	takeover	CF	ext3	**3.959**	3.959	3.958	3.966	-0.461
4	takeover	CF	squash	**4.002**	4.000	4.000	4.007	-0.417
5	takeover(total)	RD	squash	**4.588**	4.579	4.590	4.595	0.168
	dd(CF -> RD)	RD	squash	1.212	1.209	1.209	1.217	
	mount	RD	squash	0.041	0.040	0.041	0.041	
	takeover	RD	squash	3.336	3.330	3.340	3.337	

- UNIT: sec
- CF: CompactFlash / RD: RAM Disk



1. As the result of using mmap system call, bootup time is reduced by about 400msec (10% of total init time).
2. By using the takeover method, page faults are reduced to 317 times, versus 496 under the mmap method. Also, redundant page copies are eliminated. As the result, about 40msec is eliminated.
3. Squashfs is compressed ROM file system and there are some extra cost to access data, decompression and so on… But the performance is not so bad against ext3fs. Using squashfs is a good choice to reduce consumption of storage spaces.
4. Using a file system on a RAM disk is the most efficient way to increase file access performance. If the storage

device which stores the file system image is enough fast and extra RAM usage is affordable, it might be a good choice to reduce bootup time.

Case Study 2

- Status: measured
- Architecture Support:

- i386: unknown

- ARM: unknown
- PPC: unknown
- MIPS: unknown
- SH: works on SH3

Future Work/Action Items

Here is a list of things that could be worked on for this feature:

- I'm considering to implement similar file cache control using fadvise system call under 2.6 kernel.

Other resource

This project was demo-ed at the 2005 CELF Technical Conference. The picture of the poster is here:



Technical Showcase

CE Linux Forum Plenary Meeting 2005

Efforts toward improving application boot up time

Fujitsu Prime Software Technologies Limited

What is being demonstrated

Some efforts for reducing application boot up time

Target application: intent® (http://www.tao-group.com)

Characteristics of the target application:

- needs a big image file including codes, data and so on.
- loads it from a storage into memory in startup time.

Our approach:

Application side:

- front-load some data processing needed in run-time.
- change the file loading scheme.
  read(2) → mmap(2)

OS side:

customize file system cache control.

- eliminate redundant page copies.
- reduce page faults





Hardware Information

TMM1000 evaluation board (http://www.towa-meccs.com)

- SH3(7709) 133MHz, 32MB RAM, 4MB FlushROM
- running sh-linux 2.4.27

How was Linux improved

- customizing file system cache control
  - Read(2) vs. mmap(2)
    Instead of loading whole data with read system call, mapping a file with mmap system call is much faster to start up applications.
    In this case, data will be demand loading during execution.
  - Eliminating redundant page copies
    When demand loading is occurred, data from a file are kept in memory as "file system cache" and mapped to a process address space. In write access case, the file system cache is copied to a newly allocated memory page. This page can be freely modified by the process which maps it.
    Suppose a file is mapped or accessed by only one process, copying page mentioned above is redundant. We can convert file system cache to private pages immediately.
    There is a side benefit which reduces memory consumption.
  - Reducing page faults
    Process address space are accessed with read then write operations. In that case, page fault will be handled twice. With using the above page copy elimination, the second page fault can be reduced.
  - Controlling API
    It should be controlled per file or virtual memory area. The fcntl system call or mmap system call are candidates for expansion.

Patch Availability

Not decided yet

Category:

- Boot Time

> From: eLinux.org

Application XIP

Contents

Introduction

This page describes the feature Application XIP. This is a method of storing and executing applications directly from the file system, instead of first loading them into RAM. With Application XIP, the text (or code) pages of the application are never loaded into RAM. Instead, the (?? incomplete text)

Rationale

This feature is important for these reasons:

- reduction in RAM footprint
- faster first invocation
- reduction in RAM fragmentation (for non-MMU systems)
- reduced power consumption of flash vs. SDRAM

Greg Ungerer said this on the ksummit-discuss list (July 2004): "Application XIP provides a win of keeping the application code in flash even when that is shared. Can make a difference on small memory systems.

It also helps alleviate the contiguous memory problem (or memory fragmentation if you prefer) when you don't have an MMU. We need to be able to allocate a big enough contigous memory region to load the text into. Can be a problem on systems that have been running for a while and free memory is fragmented. If the application can be run XIP from flash then you at least don't need to worry about that. (This is a very real problem on small RAM systems).

The argument isn't so much RAM is cheaper than flash either. Small embedded systems will have flash and it comes in discrete sizes (1, 2, 4, 8 MB, etc). Sometimes it just makes sense to use what flash you have more effectively."

Downloads

Patch

- Code for Linear CramFS and XIP for 2.4.20 is in the base patch (base-2.4.20-8.patch) in the CELF 1.0 release. The code is not isolated into its own patch, but can be found pretty easily by grepping for CRAMFS_LINEAR. See the CELinux_040503_Release page for the source.
- Patches for Linear Cramfs feature for 2.6.9, 2.6.10, 2.6.12 and 2.6.14 are in the Patch Archive

Utility programs

How To Use

General

To use application XIP, you have to use a file system which supports it. One file system known to support it is Cramfs, described below.

Note that on an MMU-less system, the flash pages cannot be remapped to arbitrary locations in the memory space. Therefore, the text pages must be stored in flash contiguously and in order. Thus, only a file system which stores meta-data (directory nodes and index tables) separate from file data, and which allows a file to be saved as a single linear image, can support XIP under these conditions.

Linear CRAMFS Usage

The "cramfs-linear-xip" patch for kernel.org linux-2.6.9 enables a kernel to perform XIP of applications from CramFS filesystems. The code is derived from the source base of Montavista Software's products and has copyrights by Shane Nay and Robert Leslie. I'm not sure if these are actively maintained anywhere, nor whether the Montavista versions have deviated from other versions.

To use this feature...

You'll need a version of mkcramfs that does not compress files with sticky bit (chmod t) set (some versions require option -x to enable this behavior). I believe various versions are floating about, or I can look into sending a version of this as well.

Copy the resulting image to a flash partition on your board.

Build a kernel with:

```
CONFIG_CRAMFS_LINEAR=y
CONFIG_CRAMFS_LINEAR_XIP=y
```

and if it is the root filesystem:

```
CONFIG_ROOT_CRAMFS_LINEAR=y
```

When mounting the filesystem specify flags physaddr=0xNNNNNNNN, where NNNNNNNN is the physical address at which the flash partition appears. The dmesg output of the mtd probe and map drivers will show this.

For example, on my OMAP1510 Innovator mtd partition #3 is at 0x240000, as shown in the boot log:

```
Creating 5 MTD partitions on "omap_nor_cs0":

0x00000000-0x00020000 : "bootloader"
0x00020000-0x00040000 : "params"
0x00040000-0x00240000 : "kernel"
0x00240000-0x01000000 : "rootfs"
0x01000000-0x02000000 : "filesystem"
```

To mount that as root I set U-Boot bootargs as:

```
setenv bootargs console=ttyS0,115200n8 noinitrd root=/dev/null rootflags=physaddr=0x240000 rootfstype=cramfs
```

Or can mount non-root with:

```
mount -t cramfs -o physaddr=0x240000 none /mnt/mtd
```

Note that this patch bypasses the MTD subsystem and generally assumes all flash chips backing the XIP partition are wholly dedicated to serving XIP images, and that the flash chips are never in any state other than read mode (for example, no writeable partitions that are backed by the same chip). You'll need to understand the flash chip topology of your board to correctly partition the mtd devices. Also note that certain flash chips are able to independently read and write different areas and the MTD layer may support this, so the restriction might not apply to your board.

This hasn't been tested very much at this point (Nov 8th 2004) so please let us all know if any problems or suggestions.

What I will guess is that the MTD_XIP stuff (which is in linux-mtd but only for certain flash types and platforms) will eventually supplant Linear CramFS and only the XIP portions of the patch will then be needed, which might be more palatable to the kernel maintainers. I'll talk to some folks about this. CELF might then be interested in contributing to (or advocacy of) support for more flash types and platforms for MTD_XIP.

-
    - Todd Poynor, Montavista

References

- Configure Linux for XIP - Describes experience with using both Kernel XIP and application XIP.

Cramfs

CRAMFS

xip2fs

Arnd Bergman, on LKML Sep 9, 2004 with subject " " wrote: On linuxvm.org/patches, you can find a file system called xip2fs, that uses an ext2 read-only fs for XIP. The code there works only if the backing memory is a zSeries DCSS memory segment, but it should be fairly easy to port to some other low-level memory provider.

Sample Results

Here are some rough numbers: Time to run shell script which starts TinyX X server and "xsetroot -solid red", then shuts down:

Invocation	Non-XIP	XIP
First time	3.195 seconds	2.035 seconds
Second time	1.744 seconds	1.765 seconds

It was measured on a 168MHz ARM 925T TI OMAP 1510. In both cases the filesystem was in flash. Note that once the Non-XIP application pages are in RAM in the page cache, the Non-XIP case beats the XIP case (probably due to the penalty to access flash memory).

So the only performance win is on the first invocation of the application. The biggest benefit from Application XIP is from saving the RAM required for the text section of the program.

\<\>

Future Work

Here is a list of things that could be worked on for this feature:

- Cooperate with MTD to allow XIP from partitions managed by the usual MTD interfaces.
- Or at least not require the physical address to be supplied (filesystem can look it up).
- Cooperate with writeable MTD partitions on the same flash chip: suspend write/erase in progress to allow XIP reads.
- Support additional filesystems.

Sample images

Tim Riker built some sample images for a TI OMAP 1510 Innovator that use XIP kernel with XIP user space. The kernel is a Montavista kernel with XIP enabled. The filesystem uses uClibc and busybox and does a dhcp request on boot. inetd and telnetd are enabled on boot up. He uses rrload to boot with the following as kernel arguments:

```
noinitrd console=ttyS0,115200n8 root=/dev/null rootflags=physaddr=0x00260000 ip=none init=/linuxrc
```

- (1457.9 KB) XIP kernel
- (1076 KB) XIP filesystem
- (556 KB) non-xip filesystem (for comparison)

Categories:

- Boot Time
- System Size

> From: eLinux.org

Asynchronous function calls

In order to make the kernel boot faster, a set of patches was introduced by Arjan van de Ven in January 2009 to create infrastructure to allow doing some of the initialization steps asynchronously. The patches allow overlapping significant portions of the hardware delays in practice. Asynchronous function calls has been merged in mainline starting from 2.6.29. Starting from 2.6.30 the asynchronous function call infrastructure is enabled by default.

In order to not change device order and other similar observables, the patch does NOT do full parallel initialization.

Rather, it operates more in the way an out of order CPU does; the work may be done out of order and asynchronous, but the observable effects (instruction retiring for the CPU) are still done in the original sequence.

References

This work was described in the LWN.net article An asynchronous function call infrastructure

See http://lkml.org/lkml/2009/1/4/155 for the first patch in the series.

Work similar in spirit to this was done previously, but with smaller scope and apparently not mainlined.

See Threaded Device Probing

Status

This work was mainlined in kernel version 2.6.30 and is now found in the source file `kernel/async.c`

Categories:

- Boot Time
- Kernel

> From: eLinux.org

Avoid Initramfs

If you wish to improve the boot time, avoid using initramfs. Rationale is that intramfs requires copying the data twice. First the data needs to be loaded either by the boot loader or as part of your kernel image, and next the data is copied from the initramfs image to the buffer cache in the kernel.

Another disadvantage of initramfs is that the initramfs contains complete files. So if you have e.g. glibc in your initramfs it will be put into the buffer cache completely. However, only part of it will be needed as no-one uses all functions in glibc. If only part of a library or executable is used, the time to read, uncompress and copy the other parts is wasted.

Better just use the files directly from disk or flash. If your buffer cache is big enough they will stay in the buffer cache after the initial load anyway (and if your buffer cache is small you probably cannot afford to use initramfs at all).

Category:

- Boot Time

> From: eLinux.org

Bootchart

Bootchart is a tool for performance analysis and visualization of the Linux boot process. Resource utilization and process information are collected during the user-space portion of the boot process and are later rendered in a PNG, SVG or EPS encoded chart. For embedded systems several developers have tried to use bootchart to analyze boot time, but problems arose. There have been several efforts to modify bootchart to make it more useful for embedded development.

Contents

Research & Presentations

Timechart

- Tool introduced on Arjan Van de Ven's blog
- Now available in the mainline Linux kernel. It uses the 'perf' infrastructure. Hence, you need a recent enough kernel (2.6.32 or later).
- See tools/perf/builtin-timechart.c and tools/perf/Documentation/perf-timechart.txt in the kernel sources for details.

Bootchart-Lite

bootchart-lite-en.pdf Presentation by Shuuji Miyake of Fujitsu Software Technologies Limited, about bootchart deficiencies in the embedded space and ideas for fixing them.

- Not sure this is related to bootchart-lite project on Google Code, below--that was created by Fred Chien of OpenMoko.

Bootchart-lite Project Home

```
# Non-members may check out a read-only working copy anonymously over HTTP.
```

svn checkout http://bootchart-lite.googlecode.com/svn/trunk/ bootchart-lite-read-only

ubootchart

Like Bootchart Lite, an implementation of the ideas from "embootchart".

- ubootchart Project Home
- Initial blog post

EmBootchart

- Visualizing Resource Usage During Boot

  - Presentation on 'embootchart' by Matthew Klahn and Moosa Muhammad of Motorola about bootchart deficiencies in the embedded space, and a program (unfortunately never published) to fix them.

BusyBox

Busybox has a C implementation of bootchartd. It compiles to less than 40k (static uclibc i386 build). It will be available in the busybox-1.17.0 release. Please send bug reports, improvements to busybox mailing list.

- Busybox Project Home
- bootchartd applet code

```
Usage: bootchartd start [PROG ARGS]|stop|init

Options:
start: start background logging; with PROG, run PROG, then kill logging with USR1
stop: send USR1 to all bootchartd processes
init: start background logging; stop when getty/xdm is seen (for init scripts)
Under PID 1: start background logging, then execute $bootchart_init, /init, /sbin/init
This makes it possible to start bootchartd even before init by booting kernel with:
init=/sbin/bootchartd bootchart_init=/path/to/regular/init
```

Note that if you're using an initramfs, you'll need

```
rdinit=/sbin/bootchartd
```

instead of

```
init=/sbin/bootchartd
```

Note also that bootchartd tries to create a gzipped tarball, so you either need a full tar with gz support, or you need to enable FEATURE_SEAMLESS_GZ in busybox. If 'tar -z' is not working, bootchartd will not produce output.

To use

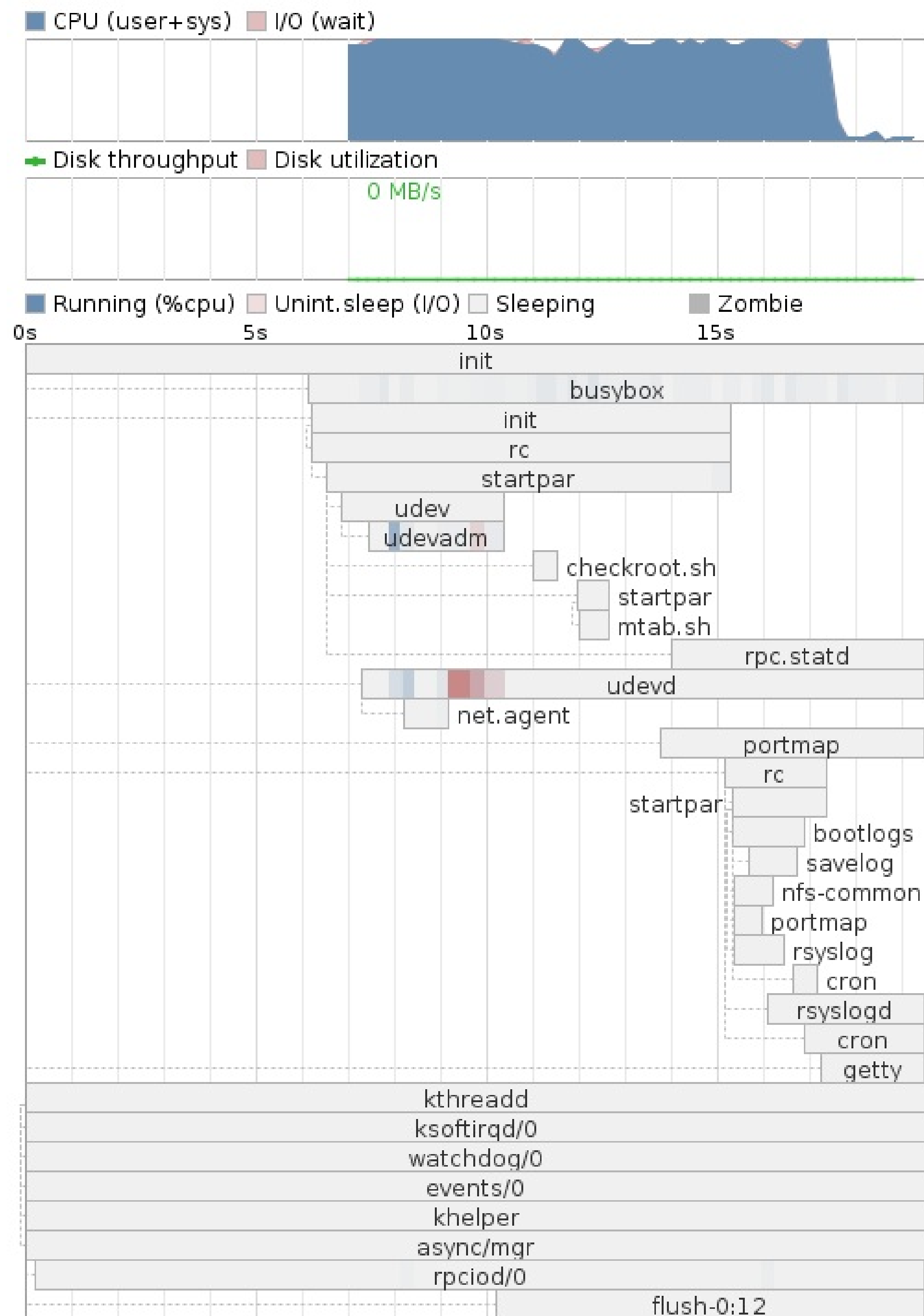
To use this, capture the bootchart information using bootchartd, then transfer the information to your host machine, and use bootchart to produce a graphic chart of the boot sequence.

The host 'bootchart' provided by Ubuntu-based hosts uses pybootchartgui to generate a graph from the collected data. However, that version of pybootchartgui does not work well with the data generated by busybox bootchartd. A newer project bootchart2 has improved pybootchartgui and included it in their sources, and this works fine with busybox bootchartd data.

After downloading bootchart2 and running 'make', you can generate a graph (bootchart.png) with:

```
 /path/to/bootchart2/pybootchartgui.py bootlog.tgz
```

An example graph, generated in a different way, is shown below:



Related projects

SystemTap Scripts

A related project is a set of System Tap scripts to provide information about boot time. See Bootprobe

Updated Fork

Bootchart has been forked and updated and can be found on github https://github.com/mmeeks/bootchart (Main changes: no java, better visualization)

Another fork (the same?) can be found here: Updated Bootchart

Categories:

- Boot Time
- Measuring
- Linux
- Bootchart

> From: eLinux.org

Boot-up Time Definition Of Terms

Definition of Terms for Bootup Times working group

Boot up The time from power on to user start (the start of user init).

Busy wait Using a timed loop to create a delay. This is often used with probing. Because the device is executing instructions in the loop, no other work can be done until the loop ends.

Cold start The time from power on to first available use.

Deferred An operation, which would occur at a specific point in the unmodified Linux boot up processing, is changed so that it can occur later in the processing.

De-serialized A set of operations, which would be serialized in an unmodified Linux environment, is changed so that the operations can be done concurrently.

Disk spinup The time required for a hard disk to reach operational speed after application of power.

Firmware The first set of machine instructions to run on the hardware after the application of power. The firmware may consist of several stages that bootstrap from one stage to the next. On an x86 machine, for example, the firmware is the BIOS. It is the responsibility of the firmware to do initial preparation of the hardware, and to load and execute the OS. This typically includes probing the hardware to discover attached devices and setting up configuration information that the OS will read during boot up.

First available use The first opportunity for actual use of the product, after a boot up, resume, or unhibernate operation.

First user experience The first visual or audio indication of activity to the user of the product, after a boot up, resume or unhibernate operation. (same as "splash")

Flash Flash memory

Hibernate The action of transitioning the machine into a no power state, in a way such that the current operating state can be restored in the future without a full initialization sequence.

Kernel decompression The action of decompressing a compressed kernel image. This stage usually includes simultaneously transferring the kernel image from persistent storage to RAM.

Kernel init The kernel initialization sequence, consisting of the period of time from kernel start to start of user init.

Kernel start The point in the boot up when the product executes the first instruction of start_kernel() routine in the kernel.

Power lost The moment that the product can no longer function because there is no more power applied to it.

Power off The moment that the machine transitions to a "losing power" state (this is not the same as the time when no more power is being applied to the CPU).

Power on The moment of application of power to the device.

Probing The act of discovering the configuration or attributes of the machine or its operating state. A probe typically consists of sending a hardware signal and waiting, for a given time, for a response from an attached bus or device.

Resume The action of transitioning the machine into a usable state from a low power state (suspend).

ROM Read-only-memory

Shutdown The action of turning the product off. The time from user power off to power lost.

Suspend The action of transitioning the product into a low power state, wherein the device cannot be used, but can be resumed quickly.

Serialized A set of operations is serialized if the operations are performed in order, without overlap or concurrency.

Splash The first visual or audio indication of activity to the user of the product, after a boot up, resume or unhibernate operation. (same as "first user experience")

Time-to-splash The time from power on to splash (or first user experience).

Unhibernate The action of transitioning the machine from hibernation into a usable state. Specifically, the usable state is the same or as close as possible to the state when hibernation was entered (eg., the same applications are running, the same files are open, etc.)

User init The time from the first instruction of the first user space program (usually /sbin/init) until the product is ready for first available use.

User power off The action or moment that the user initiates terminating the power to the device. Some devices do not use "user power off" as the primary means of "turning off" the device. Instead, a press of the "off" button may result in a suspend operation.

Warm start The time from start of resume to first available use.

XIP Execute-in-place - a method of executing code directly from ROM or Flash, without first loading it into RAM. This affects startup time, RAM and ROM footprint (size), and execution performance.

The terms used to describe boot up can refer to **events** (things that occur or start at a specific moment in time) or **time periods** (the time interval from one event to another).

Here is a list showing the sequence of main **events** during cold start: *Note:* \ means this event is optional.*

```
1.    power on
2.    firmware starts
2.1 * firmware splash
3.  * kernel decompression starts
4.    kernel start
4.1 * kernel splash
5.    user start
5.1 * user splash
6.  * RC script start
7.    application start
8.    first available use
```

Note that there may be multiple "splash" events in the system, but that the terminology here refers only to the first splash. This is also referred to in the terminology definitions as "first user experience". Splash events are highly important to diminish the *apparent* time to boot of the product. Note that the first splash event might occur at different locations in this sequence, depending on how the system is configured and what software performs the first user output. Also, even though user splash presentation is listed as event 5.1, it could occur anywhere between events 5 and 8, depending on the user-space software which performs the splash.

Here are bootup **time periods** (intervals), defined according to the above sequence of events:

- cold start: time from 1 to 8
- bootup: time from 1 to 5
- kernel init: time from 4 to 5
- user init: time from 5 to 8
- time-to-splash: time from 1 to either 2.1, 3.1, or 5.1 (depending on which is used)

Category:

- Boot Time

> From: eLinux.org

Boot-up Time Delay Taxonomy

⚠ NOTE!! - This information has not yet been transferred from the internal wiki yet.

The goal of the taxonomy is to organize the different delays into categories based on (at least) 3 different attributes:

- boot phase where boot phase is one of: firmware, kernel, user space
- magnitude
- type of delay where type is one of: systemic, hardware, busy-wait

(delay type is weird. I need to think about this more - TimBird)

Taxonomy Table

Taxonomy Table

Boot Phase	Type	Magnitude	Delay
firmware	hardware	1-8 seconds	hard disk spin-up

Category:

- Boot Time

> From: eLinux.org

Bootup Time Howto Task List

Below is a list of tasks to be worked on for Boot-up Time Reduction Howto project. If no one has signed up for a task, feel free to do so (on a first-come, first-served basis). If the link in the Technique column is red, the referenced wiki page has not been created yet.

To edit the task list, you can click on either EditText or EditTable at the bottom of this page. EditTable should be easier to use.

Please fill in the person (which can be a wiki name if you have a page on this site) or just your name. Please also provide contact information if you are not a member of the forum. If you would like to commit to a completion date, please enter that as well, in the form: YYYY-MM-DD. Finally, add any notes you think are pertinent to this task.

Individual technique pages should be created using the SingleHowtoTechniqueTemplate

Status should be one of: "not done", "started", "done", "cancelled"

Task List

Task Description	Technique	Person	Expected Completion Date	Status	Notes
Create wiki page for skipping firmware	Skip Firmware	-	-	not done	-
Create wiki page for preconfiguring PCI	PreconfigurePCI	-	-	not done	-
Create wiki page for not probing for missing devices	NoProbeMissingDevices	-	-	not done	-
Create wiki page for using a small kernel config	Small Kernel Config	-	-	not done	-
Create wiki page for reducing driver busy waits	Reduce Driver Busy Waits	-	-	not done	-
Create wiki page for threaded driver initialization	Threaded Init	-	-	not done	-
Create wiki page for loading drivers later	Load Drivers Later	-	-	not done	-
Create wiki page for eliminating some RC scripts	ReduceRCScripts	-	-	not done	-
Create wiki page for using a custom init program	Custom Init Program	-	-	not done	-
Create wiki page for reducing writes to flash	Reduce Flash Writes	-	-	not done	-
Create wiki page for disabling logging	Disable Logging	-	-	not done	-
Create wiki page for using a ramdisk during boot	Ramdisk During Boot	-	-	not done	-
Create wiki page for using a segmented file system	Segmented File System	-	-	not done	-
Create wiki page for using smaller programs	Smaller Programs	-	-	not done	-
Create wiki page for using faster memory	Faster Memory	-	-	not done	-

Categories:

- Task List
- Boot Time

> From: eLinux.org

Boot-up Time Reduction Howto

The items on this page constitute a list of existing techniques for reducing bootup times for embedded systems. Some of these items may also be applicable to desktop or enterprise systems, but that is not the focus.

For each individual item below, a separate page should exist. If it doesn't already exist, a new page should be created using the !HowtoTemplate. Links listed in red below are pages that are not created yet. To sign up for a task related to this HOWTO, please see the Bootup Time Howto Task List.

Contents

Introduction

Bootup Phases

This document divides the bootup process into 3 main phases:

- Firmware initialization phase
- Kernel initialization phase
- User Space initialization phase

User Space usually consists of a few distinct phases also:

- Initialization scripts (RC scripts, for desktop systems)
  - This is where most daemons and services are loaded.
- Graphics system initialization
- Graphical environment start
- Application initialization

Main Technique Types

Techniques presented here can be organized according to the way they try to achieve their effect. The technique can consist of:

1. speeding up the activity
2. doing the activity in parallel with other initialization tasks
3. doingthe activity later (possibly after booting is completed)
4. avoiding doing the activity at all.

In summary, each technique describes how to take an existing bootup activity and do one of:

- Do it faster
- Do it in parallel
- Do it later
- Don't do it at all

Some techniques will consist of multiple methods (such as both speeding up and doing it in parallel).

Bootup Time Reduction Technique Outline

Following is an outline of different bootup time reduction techniques, organized by the boot phase where they are applied.

Firmware Phase

Here ase some techniques for speeding up the Firmware phase of the boot sequence:

- Kernel XIP - Kernel Execute-In-Place
- SkipFirmware - Disable firmware features to eliminate diagnostics, memory counts, etc.
- Parallel HDSpin Up - Parallelize Hard Disk spinup with Kernel load
- DMA Copy Of Kernel On Startup
  - Use DMA to copy kernel from flash to RAM

Kernel Phase

The following are techniques used to speed up the initialization of the kernel:

- DisableConsole - Turn off serial console output during boot
- Preset LPJ - Use pre-set loops_per_jiffy (avoid calibrate_delay())
- Preconfigure PCI - Preconfigure some PCI bus slots
- IDE No Probe - Don't probe some IDE devices -
- No Probe Missing Devices - Disable probes for non-existent devices (including keyboards, etc.)
- Small Kernel Config - Use smallest kernel configuration possible
- Reduce Driver Busy Waits - Shorten device probes by reducing the amount of time the driver busywaits
  - A special case of this is Short IDE Delays, with IDE driver delays
- Threaded Init - Perform threaded initialization - replace driver busywaits with yields
  - A special case of this is IDE Preempt, with IDE driver busywaits
- Load Drivers Later - Use modules where possible to move driver initialization later in the boot sequence

User Space Phase

The following are techniques for reducing the bootup time for user-space programs:

- Application XIP - Execute-In-Place for applications and librarys
- Reduce RC Scripts - Eliminate unneeded RC scripts
- Custom Init Program - Use a custom initialization program
  - (This is a special case of eliminating unneeded RC scripts)
- Optimize RC Scripts - Optimize RC script execution
- Parallel RC Scripts - Execute RC scripts in parallel, instead of in sequence
- Pre Linking - Avoid overhead of runtime link fixups during first program/library load
- Reduce Flash Writes - Reduce writes to flash. (In particular, perhaps you want to "disable the date of last access" with noatime [1][2][3][4][5][6])
- Disable Logging - Turn off logging to stable storage
- Faster File System - Use faster file system

- Ramdisk During Boot - Use RAMDISK during boot
- Segmented File System - Use a segmented file system to avoid interference between reads and writes

General Reduction Techniques

Some reduction techniques don't apply to a specific boot phase, but are general methods to reduce bootup time. These are listed here.

- Smaller Programs - Use smaller kernel and programs for faster load times
- Faster Memory - Use faster system memory to increase load and initialization performance

Table of Reduction Techniques

The following table summarizes the various techniques listed in this document.

Acronyms and Terms

Technique	Boot Phase	Description	Observed reduction	Notes
Kernel XIP	Firmware	Kernel Execute-In-Place - avoids kernel copy and decompression time	250 ms	causes runtime performance loss
Skip Firmware	Firmware	Skip firmware probing features, like memory check, bus probing, and device detection, etc.	??	Linux re-probes busses and devices anyway, so this is usually waste of time
Parallel HDSpin Up	Firwmare	Start hard drive spin up before loading kernel	??	Not possible if the kernel is loaded from hard drive.
DMA Copy Of Kernel On Startup	Firmware	Use DMA to copy kernel from flash to RAM	180 ms	.
Preset LPJ	Kernel	Use a hardcoded loops_per_jiffy value to avoid cost of calibration.	250 ms	.
No Probe Missing Devices	Kernel	Avoid probing for non-existent keyboards and other devices	??	.
Small Kernel Config	Kernel	Reduce kernel size and length of code paths, thereby reducing execution overhead	??	.
Disable Console	Kernel	Turn off output to serial console during boot	250 ms	.
Preconfigure PCI	Kernel	Preconfigure PCI bus slots on kernel command line	??	Is this even possible?
Load Drivers Later	Drivers	Move drivers to modules and load them later in boot sequence.)	??	Only works for drivers that can be loaded as modules late in the boot cycle.
IDE No Probe	Drivers	Use "noprobe" on kernel command line for IDE driver	3 sec.	Depends on hardware present
Reduce Driver Busy Waits	Drivers	Reduce the length of driver busy waits	??	.
Short IDE Delays	Drivers	Reduce length of IDE initialization delays	5 sec.	May be dangerous, depends on hardware
Threaded	Drivers	Replace busywaits in drivers with	??	Only adds value if driver can be parallelized with

Init		yields		some other init activity.
IDE Preempt	Drivers	Replace busywaits in IDE drivers with yields	250 ms (decreased non-preemptibility)	Already fixed in 2.6
Reduce RC Scripts	RC scripts	Eliminate unneeded init scripts	3 sec.	Depends on required scripts
Parallel RC Scripts	RC scripts	Start init scripts in parallel	??	.
Defer RC Scripts	RC scripts	Defer some init scripts to later in boot cycle	??	.
Optimize RC Scripts	RC scripts	Use busybox, smaller shell, builtins, adjusted scripts	3 sec.	Depends on required scripts
Custom Init Program	RC scripts	Use custom initialization program (eliminating RC scripts altogether)	10 sec.	requires long-term maintenance of the program
Application XIP	User Space	Use Execute-In-Place for applications and libraries.	??	Requires uncompressed file system. Application performance may be reduced.
Segmented File System	User Space	Keep read-only data separate from writable data in flash storage	??	.
Reduce Flash Writes	User Space	Avoid writes to flash memory	??	.
Ramdisk During Boot	User Space	Keep writable files in RAM, and write them to flash after boot	??	.
Smaller Programs	User Space	Use smallest programs and configurations possible	??	Reduces program load time. It may increase cache hits.
Faster Memory	General	Use faster memory	??	.

Potential Techniques

Here is a list of potential techniques that have not been tried yet, to our knowledge:

- Use different, faster, firmware
- Cache results of find and grep during RC scripts
- Partial XIP (this is a current project of the WG)

Category:

- Boot Time

> From: eLinux.org

Bootup Time Spec

This is the specification of bootup time technologies and features, of the Bootup Time Working Group of the CE Linux Forum.

```
#noprint
[[TableOfContents3]]
#noprint
```

Contents

Introduction

The specifications of the Bootup Time Working Group deals with reducing the time required to boot a Linux kernel in a consumer electronics products. The purpose of this specification is to define requested or required features of Linux which improve the bootup time of the system for such products. Also, this specification mentions features which, over the long term, will assist developers in measuring and enhancing the bootup time for their systems.

While suspend, resume, and shutdown times are also within the scope of the Bootup Time Working Group, this version 1.0 of specifications does not include any technology in support of reductions in these areas. Reductions in those areas will be covered in future versions of the specification.

Rationale

Users expect to be able to use their CE products very soon after turning them on. Linux, as configured and used for desktop and server systems, exhibits long bootup times - on the order of 30 seconds to a few minutes. The technologies mentioned here (while small in number, in this first release), represent a few mechanisms which can be used to reduce bootup time.

Terminology

The following table presents terms used in this specification related to bootup time technology and features.

Work In Progress

The following item is currently being worked on, but is not ready for publication yet.

Category:

- Boot Time

> From: eLinux.org

Bootup Time Task List

Below is a list of tasks to be worked on in the Bootup Time Working Group. See also the Bootup Time Howto Task List

Instructions

If no one has signed up for a task, feel free to do so (on a first-come, first-served basis).

To edit the task list, you can click on either EditText or EditTable at the bottom of this page.

Please fill in the person (which can be a wiki name if you have a page on this site) or just your name. Please also provide contact information if you are not a member of the forum. If you would like to commit to a completion date, please enter that as well, in the form: YYYY-MM-DD. Finally, add any notes you think are pertinent to this task.

If you need to create a page with more detailed information about the task, please do so and link to that page either in the Task Description or the Notes field.

Status should be one of: "not done", "started", "done", "cancelled"

Task List

Task Description	Project	Person	Expected Completion Date	Status	Notes
Make this task list	WG process	Tim Bird	2004-01-12	done	This task is recursive :)
isolate KFI patch from CELinux tree	KFI	-	-	not done	-
check for IDE noprobe bugfix in 2.6 kernel	noprobe	-	-	not done	-
port printk patch to 2.6 kernel	printk	-	-	not done	-
examine bootsplash project and see if it is interesting	bootsplash	-	-	not done	-
create a table of reduction techniques, and ask people to sign up for each page. See Bootup Time Howto Task List	howto	Tim Bird	-	done	-
make wiki pages for each workgroup project, spec, and implementation	process	-	-	not done	-
Document KFI usage (triggers and filters)	KFI	Todd Poynor	-	not done	-
write sample specification for avoiding calibrate_delay()	LPJ	Tim Bird	-	not done	-
create outline of kernel init sequence	howto	-	-	not done	-
create core group	process	All	-	not done	-

Categories:

- Task List
- Boot Time

> From: eLinux.org

Bootup Time Things To Investigate

This page has a list of items which may be worth investigating in the future.

If you see something that creates a long delay (e.g. over 250ms) then add it here and maybe some day we will try to investigate it.

netconsole extra delay for bogus network carrier detect

Tim Bird saw this on the linux-tiny list: (see http://www.selenic.com/pipermail/linux-tiny/2004-August/000015.html)

Network devices that say they've got carrier detect instantly at boot are assumed to have bogus carrier detect and netconsole waits a bit to assure the line is up and ready to go before proceeding. The timeout mentioned below could probably be shrunk.

```
>eth0: NE2000 found at 0x300, using IRQ 5.
>> Universal TUN/TAP device driver 1.5 (C)1999-2002 Maxim Krasnyansky
>> netconsole: device eth0 not up yet, forcing it
>> eth0: driver changed get_stats after register
>> netconsole: carrier detect appears flaky, waiting 10 seconds
>>                                            ^^^^^^^^---------this delay seems long
```

Category:

- Boot Time

> From: eLinux.org

Bootup Time Working Group

This page has references for various resources for use by the Bootup Times Working Group.

Table of Contents:

Contents

Working Group Information

Charter

The Bootup Time Working Group shall work to minimize the activation and deactivation times of Linux systems. This includes making improvements in cold start bootup times and shutdown times, as well as improvements in the speed of suspend and resume operations. The Working Group shall establish requirements for Linux systems and sub-systems in order to accomplish this goal of timely activation and deactivation. Also, the Working Group will evaluate and recommend technical solutions and implementations which accomplish this goal.

Scope

The scope of this Working Group includes firmware, operating system kernel, and user space issues. It may include work to accelerate device initialization by means of coordination between the firmware and the OS, work inside the OS to reduce the time to initialize kernel sub-systems and device drivers, and work to increase application startup speed. Also, the Working Group may specify, evaluate or recommend instrumentation and tools for analysing bootup and shutdown times.

The Working Group will not consider compiler technologies related to this issue.

Current Tasks

See the Bootup Time Howto Task List

Documents and information

- Boot-up Time Definition Of Terms
  - defintions of terms used by the working group
- /\ - *no content yet* - Boot-up Time Delay Taxonomy
  - list of delays categorized by boot phase, type and magnitude
- Presentation: - Reducing Startup Time in Embedded Linux Systems This document is a presentation that was prepared based on existing bootup time reduction techniques.
- Kernel Instrumentation - lists some known kernel instrumentation tools. These are of interest for measuring kernel startup time.
- Filesystem Information - information about bootup times with various file systems

Current Projects

Boot-up Time Reduction Howto - this is a project to catalog existing bootup time reduction techniques. Work on this project is under way. The wiki will serve as the primary repository of information gathered for this project.

Specifications

- Timing_APISpecification
  - requirements (and specification?) for a simple API to support bootup timing measurements
- see also Instrumentation_API for some background research on this API
- Calibrate Delay Avoidance Specification - avoiding the cost of calibrate_delay()
  - Done - see Preset LPJ
- IDE_No_Probe_Specification
  - force kernel to observe the IDE "noprobe" command line option
- IDE_Preempt_Specification
  - change IDE busywaits into preemptible timeouts
- Kernel_XIP - support Execute-In-Place for the kernel.

Implementations and/or patches

Measurement Systems

- Printk Times - simple system for showing timing information for each printk
- Kernel Function Instrumentation
  - more complete system for reporting function timings in the kernel (The patch for this has not been isolated, but it's currently in the CELF tree)

Patches for Reducing Bootup Time

- Preset LPJ - Allow the use of a preset loops_per_jiffy value
- IDE No Probe - Force kernel to observe the ide\=noprobe option
- IDE Preempt - Make IDE driver init busywaits preemptible
- Kernel XIP - Allow kernel to be executed in-place in ROM or FLASH (code is not isolated yet, but is in the CELF source tree)

External projects

De-serialized user-space service startups (RC scripts)

- IBM article on on using Makefile techniques to express dependencies between services and support parallel service start. See http://www-106.ibm.com/developerworks/linux/library/l-boot.html?ca=dgr-lnxw04BootFaster

- Richard Gooch project to rewrite boot script system from scratch. Eliminates lots of BSD and SYS V-isms, and introduces dependencies. See http://www.atnf.csiro.au/people/rgooch/linux/boot-scripts/
- Serel project - for parallelizing service startup. Commands are inserted into RC scripts to cause needed services to start (based on XML database of dependencies). See http://www.fastboot.org/
- LSB specification for comments in RC Scripts which allow parallization. See http://www.linuxbase.org/spec/refspecs/LSB_1.1.0/gLSB/initscrcomconv.html

Kexec

- Kexec is a system which allows a system to be **rebooted** without going through BIOS. That is, a Linux kernel can directly boot into another Linux kernel, without going through firmware. See the white paper at: http://devloper.osdl.org/rddunlap/kexec/whitepaper/kexec.pdf
- here's another Kexec white paper: http://www-106.ibm.com/developerworks/linux/library/l-kexec.html?ca=dgr-lnxw04RebootFast

Pre-Linking and Lazy Linking

- see this excellent paper for an overview of dynamic linking issues: http://www.cis.upenn.edu/~mwh/papers_DB/ieee_computer97.pdf

Others

- http://www.bootsplash.org/
- http://www.linuxdevices.com/news/NS5907201615.html
    - any FSM Labs folk have pieces of this?

Categories:

- Boot Time
- CE Linux Working Groups

> From: eLinux.org

BusyBox

Busybox is a program with small memory footprint, designed to replace a suite of otherwise large Linux utilities. It is commonly used in embedded Linux systems. Over 58 projects are listed on the busybox projects page.

The text segment is only loaded once. Subsequent calls to subprograms use the same text segment, saving memory.

Additionally, it is highly configurable. Only the required applets are compiled in reducing storage requirements.

See http://www.busybox.net/

A good article on busybox is at: http://www-128.ibm.com/developerworks/linux/library/l-busybox/

Categories:

- Linux
- Software

> From: eLinux.org

Deferred Initcalls

Contents

Introduction

An "initcall" is the initialization function for a module which is statically linked into the Linux kernel. Running of initcalls can consume a large amount of time during bootup. However, some modules do not need to be initialized so early, in order for the embedded device to become usable for its primary purpose. These initcalls can be run later, after first boot (and possibly first use) has completed.

For example, many digital cameras have USB buses, which need to be initialized in order for the camera to be used as a mass storage device (to download pictures to a desktop computer). However, the USB buses do not need to be initialized for the primary use of the camera, which is to take pictures. In order to be able to take a picture as soon as possible after booting, initialization of the USB system in the kernel can be deferred until after the primary boot sequence for the camera.

Description

Using a short patch (available for kernel version 2.6.27) it is possible to avoid running certain initcalls at bootup time. The way this patch works is that the developer selects some initcalls that they want to defer and modifies the declaration of those initcalls.

When the system is booted, those initcalls are not called in their normal sequence. When the system is done with primary boot, an application from user space triggers the calling of the deferred initcalls, using a flag in the /proc filesystem.

When the flag is set, the deferred initcalls are run, and the kernel memory for the "init" segment is finally freed.

How to use

To use deferred initcalls, first you need to identify the modules that are not required for the primary functionality of the product, and which are taking too long to initialize. (See Initcall Debug for this.)

Using the example above of the digital camera and USB, you could identify uhci_hcd_usb and ehci_hcd_init as two initcalls that could be deferred.

Change the module init routine declaration for the initcalls you wish to defer. This is done in the Linux kernel source code. For example, change:

```
module_init(foo_init)

deferred_module_init(foo_init)
```

Modules marked like this are not initialized during kernel boot

After main init, do:

```
cat /proc/deferred_initcalls
```

This will cause the kernel to run all deferred initcalls. Also the .init section memory is freed by kernel. The contents of /proc/deferred_initcalls will return 0 if deferred initcalls were not yet run, and 1 otherwise on subsequent reads.

deferred USB initcall example

As a test, on an X86 desktop system, I deferred the initialization of the USB subsystem on a 2.6.27 kernel, by using deferred_module_init on the functions: ehci_hcd_init and uhci_hcd_init

This resulted in a total times savings of 530 milliseconds, during the kernel boot phase. (Of course, this time was used subsequently when the deferred initcalls were triggered later on.)

Specifially, I changed:

```
module_init(ehci_hcd_init)
```

to

```
deferred_modle_init(echi_hcd_init)
```

and

```
module_init(uhci_hcd_init)
```

to

```
deferred_module_init(uhci_hcd_init)
```

Patch

Here is the main deferred initcalls patch for 2.6.26, 2.6.27: Media:Deferred_initcalls.patch

For 2.6.28 the forward-ported patch is here: Media:Deferred_initcalls-2.6.28.patch

For 3.10 the patch is here: Media:0001-Port-deferred-initcalls-to-3.10.patch

Here (inline) is a patch showing modification of USB and IDE initcalls to be be deferred initcalls: (This patch is also available downloadable as: Media:Defer-usb-and-ide-initcalls.patch)

```
commit e7a5b8bb6a5d04054dec1e85d53bbe115059d0d0
Author: Tim Bird <tim.bird@am.sony.com>
Date:   Fri Sep 12 11:35:58 2008 -0700

    Use deferred_module_init on long-probing IDE and USB modules.
    These modules were taking about 700 ms and 400 ms, respectively
    to initialize.  On many embedded systems, these initializations
    can be done after major boot activity is completed, with no
    loss of functionality.

diff --git a/drivers/ata/ata_piix.c b/drivers/ata/ata_piix.c
index e9e32ed..cb2ebf3 100644
--- a/drivers/ata/ata_piix.c
+++ b/drivers/ata/ata_piix.c
@@ -1494,5 +1494,5 @@ static void __exit piix_exit(void)
    pci_unregister_driver(&piix_pci_driver);
 }

-module_init(piix_init);
+deferred_module_init(piix_init);
 module_exit(piix_exit);
diff --git a/drivers/usb/host/ehci-hcd.c b/drivers/usb/host/ehci-hcd.c
index 8409e07..44a8340 100644
--- a/drivers/usb/host/ehci-hcd.c
+++ b/drivers/usb/host/ehci-hcd.c
@@ -1107,7 +1107,7 @@ clean0:
 #endif
    return retval;
 }
-module_init(ehci_hcd_init);
+deferred_module_init(ehci_hcd_init);

 static void __exit ehci_hcd_cleanup(void)
 {
diff --git a/drivers/usb/host/uhci-hcd.c b/drivers/usb/host/uhci-hcd.c
index 3a7bfe7..9c27ef0 100644
--- a/drivers/usb/host/uhci-hcd.c
+++ b/drivers/usb/host/uhci-hcd.c
@@ -999,7 +999,7 @@ static void __exit uhci_hcd_cleanup(void)
    kfree(errbuf);
 }

-module_init(uhci_hcd_init);
+deferred_module_init(uhci_hcd_init);
 module_exit(uhci_hcd_cleanup);

 MODULE_AUTHOR(DRIVER_AUTHOR);
```

Categories:

- Boot Time
- Kernel

> From: eLinux.org

Disable Console

Contents

Description

You can save time during kernel bootup by disabling the console output. The easiest way to do this is to use the "quiet" option on the kernel command line (described below).

Printk output is usually directed to a serial port or a VGA console during bootup. By disabling console output, the time taken to output the characters (and perform things like software scrolling of the display buffer) is eliminated.

How to implement or use

To disable console output during kernel bootup, use the "quiet" option on the kernel command line.

To do this, just put the word "quiet" (without the quotes) in the kernel command line, with no other arguments. This will suppress printk output during booting. Note that printk messages are still buffered in the kernel and can be retrieved after booting using the "dmesg" command.

Expected Improvement

This saves time during kernel bootup by suppressing printk output. Printk output delays depend on a number of factors, but in the use cases cited below, the savings were in the range of a few hundred milliseconds.

With a serial console, the time to output characters is dependent on the serial port speed. However, with a VGA console, the time to output the characters is dependent on the speed of the processor. Therefore, the slower your processor, the more savings you will gain from this technique.

Resources

Projects

None

Specifications

None

Patches

None

Case Studies

Case 1

Hardware	KMC SH board, using VGA console
Kernel Version	CELF-1 (040126)
Configuration	relatively small configuration (*details not available*)
Time without "quiet" option	637878 usec
Time with "quiet" option	461893 usec
Time savings	176 milliseconds

From data submitted by Maruyama Kiyoyasu of Mitsubishi.

Case 2

Hardware	TI OMAP board, using serial console
Kernel Version	CELF-1 (040126)
Configuration	Kernel booted with XIP, CRAMFS root file system, with preset-LPJ
Time without "quiet" option	551735 usec
Time with "quiet" option	280676 usec
Time savings	271 milliseconds

From data submitted by Noboru Wakabayashi of Hitachi.

Categories:

- Boot Time
- Kernel

> From: eLinux.org

DMA Copy Of Kernel On Startup

Contents

Description

This section describes the technology of DMA copy of kernel on startup. It's necessary to copy binary images, such as a kernel image, file system images, and so on, from ROM to RAM on bootup if XIP isn't used. In this case, using DMA transfer is very efficient to save the time and CPU resources.

Rationale

This feature is important to CELF because it reduces bootup time significantly.

Resources

Projects

None.

Specifications

None.

Downloads

Patch

Sorry but there is no available patch at this time.

Utility programs

None

How To Use

Sample Results

Case Study 1

Jyunji Kondo (of Fujitsu Prime Software Technologies) measured the result of using DMA kernel copy on FR-V processor.

Hardware	Fujitsu FR-V Design Kit for FR450 core
Kernel Version	2.6.6
Configuration	- DMA kernel copy enabled/disabled - Preset LPJ enabled/disabled - Avoiding printk messages enabled/disabled

Here is the graphic chart which illustrates the elapse time of each startup stage.

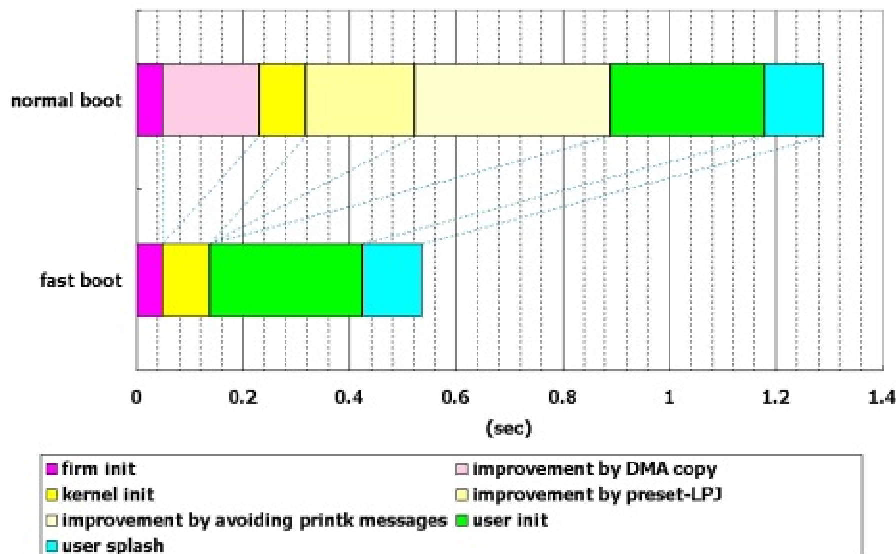


Figure 1: Comparison of normal and fast boot

⚠ Hardware initialization consumes about 200 milliseconds before firmware init.

And here is the table which shows the actual performance number.

Table 1: Elapse time of each stage (in milliseconds)

-	firm init	kernel init	user init	user splash	total
normal boot	229	660	290	112	1,291
fast boot	49	88	287	113	537

In the normal boot case, the kernel image whose size is 1.7MB around is copied from ROM to RAM by CPU at firmware init. On the other hand, DMA copy is used in the fast boot case and it reduces about 180 milliseconds.

It owes much to the potentiality of FR-V processor, but it's worthy of consideration to use such a hardware assist feature for reducing bootup time.

Case Study 2

Status

Status - measured not started, researched, implemented, measured, documented, accepted

- Architecture Support:

  (for each arch, one of: unknown, patches apply, compiles, runs, works, accepted)

  - i386: unknown
  - ARM: unknown
  - PPC: unknown
  - MIPS: unknown
  - SH: unknown
  - FR-V: works

Future Work/Action Items

Categories:

- Boot Time
- Bootloader

> From: eLinux.org

Fast Booting Translation

瞬間起動に関する一考察

Contents

はじめに Introduction

```
近年、デジタル情報機器の普及に伴い、
In recent years, with the spread of digital information equipment,
起動時間の大幅な短縮に関する研究が行われております。
research is being conducted to drastically reduce the startup time of products.
最近の高機能化競争に伴い、起動時間は非常に長くなっているので、
Startup time is very long...(recent performance race??)
その短縮が重要なのは当然です。
Short course is very important.
さらに、最近は、MRAMなどをつかい、
Moreover, recently, MRAM (has recently been introduced)
まったく瞬間的に起動する技術もあります。
This technology does not start instantaneously.
これは、単に起動が高速になるだけではなく、
this is not just simply a fast start
高速な起動により、こまめな電源切断の負担を減らすため、
It is a fast start, while frequently cutting power to reduce the burden
とりわけ、モバイル型デジタル情報機器にとっては、
In particular, mobile devices for digital information
消費電力の削減になります。
need reduced power consumption.
また、据え置き型の場合でも、待機電力の削減にも
The deferred-type cases, the reduction of standy-by power
結びつきます。
(???)
以上を考えれば、
Given the above
利便性と地球環境の双方により
and the convenience of both the global environment
重要な研究課題といえます。
this is an important research topic.
```

手法１ スリープモード 1. Sleep techniques

```
常に電流を流しておく方法を、仮にスリープモードと呼びます。
One method of faster starting is to always keep the current flowing, even if sleep is called.
この手法は、ビデオデッキなどでは一般的です。
This technique is used for appliances such as VCRs.
この場合、起動は非常に高速になりますが、欠点が二つあります。
In this case, there is a very fast start, but there are two drawbacks:
```

１.待機電力が増える。

```
1. Standby power utilization is increased.
```

２.リブートは高速化されない。

```
2. A full reboot of the product is still slow.
```

手法２ サスペンド 2. Suspend methods

```
固定的なサスペンドイメージを用い、高速に起動させる。
This method consists of using a permanent suspended image, with a fast start.
・・・って、サスペンド法で用語はいいのか？
Is suspend (the law?) a good term?
スナップショット法、とかの呼び方もあるでしょうが、
This is sometimes referred to as snapshot boot.
どうしましょうか。
How is this done?
で、このML的には、ソニーの神長氏がやってますし、
The ML, however, Sony seems to be doing this OK.
商用では、トライピークス社がやっています。
欠点は以下。
The following are drawbacks:
１． ブートのたびに、同じ状態に戻ってしまう。
1. Every time you boot, you return to the same state
２． 同一ハードでないと動かない。
2. And not the same hard work (??)

[２] の欠点のため、通常のPC等には不向きで、
情報家電向け手法といえます。
This drawback makes this technique unsuitable for
PCs and consumer electronics devices.
```

手法３ CD-ROM ブートの高速化 3. An approach for faster booting of CD-ROM

```
手法はいろいろあります。
There are lots of techniques.
手前味噌ですが、弊社の MACH BOOTが最速でしょう。
The fastest one we know of, which our company wrote, is MACH BOOT.
ちなみに、もっと高速化できるんですが、
Incidentally, there is more speed
皆さんの反応からすると、
but the reaction from folks
高速化よりも機能性などの追及のほうが重要っぽいです。
利点：
・ほぼ完璧なPnPを瞬間的に実行。
・CD-ROMなので、プレスによる大量配布が可能。
```

手法４ 不揮発性メモリの利用

```
MRAM/FeRAM などとロジック回路を組み合わせ、
突然電源を落としても今までの状態が保てるようにする。
最近出てきた方法。
というか、これの実用化の可能性が見えてきたので
この駄文を書いている。
ただし、その結論は、「あまり意味はない」となるんですが。
```

瞬間起動の使用状況

```
ここで、「使用状況」という用語を使うのが正しいかわかりませんが、
ようは、どういったときにどういった連中が何のために使っているか、
というような極めてマーケティング的な話です。
で、これ、マーケティング用語として正しいんでしょうか。
とりあえず正しいとして話を続けます。
とにかく、このマーケティング的な話が重要なんです。
```

瞬間起動の使用状況 デジカメ編

```
たとえば、デジカメを考えます。
シャッターチャンスを逃さないためにも、瞬間起動は重要だと思います。
でも、べつに0.01秒で起動する必要性はないわけですよ。
すくなくとも、全機能が0.01秒で起動する必要性はない。
デジカメの場合、CCDの出力がファインダー液晶に写るようになる
（なんていうんだ？）までの時間は早ければ早いほどいい。
0.01秒とかなら最高。
でも、マスコミの記者とか、特殊な職業じゃないかぎり、
シャッターを押せるまでの時間がそこまで短い必要性はない。
1秒かかってもOK。
ましてや、全機能が瞬間的に使える必要性なんて全然ない。
コンフィグのメニューなんて、起動に10秒かかってもOK。
ようは、ファインダー出力の部分だけ瞬間的に起動するようにすればいい。
で、これだけど、たとえその部分に何らかのマイコン制御を掛けていても、
その起動なんてすげぇ簡単だから、
瞬間起動なんて誰でも実装できる。
ようは、実際の使用状況における瞬間起動は、
ぜんぜん難しくないんだ。
```

瞬間起動の使用状況 HDDレコーダー編

```
いま、もっとも瞬間起動が求められているのはこれかも。
ようは、いまのHDDレコーダーって、実は、
Linux内蔵のある種のマルチメディアパソコン（笑い）なんだけど、
とにかく起動が遅い。
1分とか掛かるんだよ、信じられない。
原理的には、お皿の回転待ちとかを考えても、
VHSのローディングより高速にできるだろ？
第一さ、HDDレコーダーってのは、
決定的瞬間を逃さず録画できる。
見たいシーンがすぐ見れる。
そういったことがもともとウリだったんだから。
じゃあ、何秒で起動させればいいか。
1秒か2秒でしょう。
これ以上高速化しても、あまり意味はない。
それとも、0.01秒起動だけど3割高い、
というHDDレコーダを買いますか？。
```

情報家電

```
ここでいう情報家電というのは、
たとえば、フラッシュROMに独自Linuxデスクトップ、Firefox、OpenOffice
などがプレインストールされているIBM-PCであり、
そのプレインストールソフトでの利用しか前提とされていない点を除けば、
通常のIBM-PCと（ほぼ）同等のものをさします。
別の言い方をすると、昔のワープロ専用機の進化系です。
まあ、企画案だけはどこのメーカーでも出まくってるんでしょう、多分。
で、これなんですが、1秒か2秒でブートすれば十分でしょう。
むしろ重要なのは、電源OFFにかかる時間で、
1秒でサスペンド（ハイバネ）して電源OFFとかだとおいしいですな。
ようは、文書の保存とか考えなくても、
極端な話、いきなりコンセントを引き抜いても、
キャパシタで1秒間持たせて、そのあいだにサスペンドするとかさ。
これはウケるよ。
```

PDA

```
ザウルスとかiPhone みたいなのね。
これも、1秒か2秒で十分。
細かいことを言うと、
画面のレジュームだけは先にやるといいかも。
ようは、前回電源OFFのときの画面が、
電源ONの瞬間に戻るとそれはそれでいいかも。
なお、やり方はいろいろあるけど、
突然電源が切れるようにしてくれ！。
しかも、電源ボタン長押しとかじゃイヤだ。
スライドスイッチでオンオフできるようにしてくれ！！！。
スライドスイッチをやたらカチャカチャさせても
何の問題もないようなやつがいい。
当然、文書作成中に電源を切っても、
作成中の文書は消えないのは当然。
できれば、すべての状況がサスペンドされているようなやつね。
```

携帯電話

```
これもやはり1秒か2秒。
本当は、受け待ち状態への移行が瞬間的だと
省電力とかとも絡んで面白いんだけど、
電波を掴むのに必要な時間も考えると、
OSのブートとはあまり関係のない話。
ただし、電話番号の入力だけは瞬間的にできるといいかも。
電源オンの0.01秒後から入力可能、みたいなのね。
```

すべてを高速化する必要性はない

```
以上の議論でわかるように、
0.01秒などの瞬間起動がすべての場合で必要なわけではない。
どの程度の起動速度が必要なのかは、
ユーザニーズとコストのバランスを睨みながら
慎重に決定するとともに、
同じ起動速度をすべての部分に適用するのではなく、
部分部分で起動速度の目標を個別に設定する。
```

起動速度の高速化の限界

```
もちろん、MRAMなどを使えば、まったく瞬間なのだが、
たとえば、サスペンド法などを使っても、
実質的には非常に高速に起動させることは可能。
なぜなら、サスペンド法を使った場合の起動時間は、
以下の３つに使われている。
１、ハードウェアの安定化
２、ハードウェアの初期化
３、サスペンドイメージのロード
[１]は、ようは、コンデンサがチャージされて
パワーオンリセットされて、、、という時間ね。
（これ、なんていうんだ？）。
ここに関しては、べつにMRAMを使っても同じ時間はかかる。
[２]だが、これをほぼゼロにすることは問題ない。
単に、設定レジスタに適当に値を入れていくだけ。
IBM-PCとかでこれが妙に時間がかかるのは、
PnPがあるうえに、奇妙なワークアランドの関係で
意味不明のウェイトを入れまくってるから。
そのあたりを無視し、
特定ロットだけ動けばOK、とばかりにウェイトを外せば、
いまでも非常に速くなる。
[３]だけど、これだっていくらでも高速化可能。
正直、たとえば、
いまのIBM-PC＋いまのCD-ROMドライブでも、
たとえば、Windowsデスクトップを5秒以内に
起動することは実は可能。
それも、たとえば Vistaだって起動可能。
というか、いまのOSの起動は遅すぎる。
ましてや、ちょっとH/Wをイジるとともに、
ブートデバイスをフラッシュROMにすれば、1秒以内で起動できる。
Windowsが1秒で起動だよ。これ、遅いかな？
```

スリープモード

```
確かに、スリープモードは待機電力を使う、という問題点がある。
しかし、極限まで待機電力を少なくした素子を開発することは可能。
MRAMを使った瞬間起動可能素子の開発と比較し、
どちらが容易かは議論がある。
もし、待機電力が実質ゼロの素子があれば、
MRAM方式はよりいっそう意味を失う。
```

MRAMを用いた瞬間起動

```
確かに、この方法は、実質的にまったくゼロ時間での
電源ONおよび電源OFFを可能にし、
これは、たとえば、電源スイッチのないノートPC、
といったまったく新しい商品の招来すら可能で、
非常に興味深い技術といえる。
しかし、この技術を利用することは、
新技術に基づく、
低機能かつ高価な部品を利用することになる。
この傾向は、いかにMRAM技術が進化しても、
ほぼ永遠に変わらないものと思われる。
また、当面の間、技術的に不安定であって、
量産上の問題点などが予期される。

そのような問題点を差し置いてでも利用すべきか
否かであるが、筆者（岡島）は非常に否定的な意見を持っている。

理由は、MRAMを利用しないと達成できないような
起動速度が要求されている部分が、実際の使用においては
ごく一部に過ぎないことである。
そして、その部分も、MRAMの利用だけが唯一の対応策ではなく、
そのほかの策でも高速化が可能な場合がほとんどだと思われる。

よって、MRAMによる高速化は、現在の時点での予測としては、
非常に小規模な利用にとどまると思われる。

なお、もし、逆に、MRAMによる瞬間起動が大規模に利用されるとすれば、
これを前提とした、画期的機器が現れたときであろう。
具体的には、上でも述べたが、そもそも電源スイッチという概念の存在しない
ノートPCなどである。
しかし、そのためには、内部の素子だけでなく、
液晶もメモリ一性がありなおかつ反応速度が速くなおかつ高コントラスト、、
みたいなまったく新しいものが必要であるし、
また、そもそも、OSのスケジューリングもアプリの内部構造も、
そしてユーザインターフェースも・・・
すべてを変えないといけない。
ようは、あまりにも大規模な開発であって
正直にいえば非現実的としか思えない。
多分、なにかには応用されるだろうが、
もっと小規模なニッチ的商品ではないのか？
```

まとめ

```
サスペンド法などの従来手法による高速化だけでも、
十分な速度が得られる。
MRAMなどによる高速化は、非常に特殊な場合のみにとどまるであろう。

-------------
Further reading:
Check Nikkei Electronics (magazine) Jul 31 2006 issue.
以上の話は、日経エレクトロニクス2006年7月31日号(931号）にも
載っています。
例の台湾大手EMS、ホンハイの特集号なんで、
そういった観点からも面白いかも。
```

Category:

- Boot Time

> From: eLinux.org

Fast Kernel Decompression

This page has notes about faster kernel decompression.

Contents

Description

Currently, the method used to compress the kernel is gzip. However, other compression and decompression methods exist which may allow improvements in kernel decompression (and hence startup) performance.

This page documents Sony's investigation of UCL compression/decompression performance, for possible use in speeding up bootup time on an embedded device. In our testing UCL decompressed a sample file system image 43% faster than gunzip, and a sample kernel image 28% faster than gunzip.

From the UCL web page, it states:

- UCL is written in ANSI C. Both the source code and the compressed data format are designed to be portable across platforms.
- UCL implements a number of algorithms with the following features:
  - Decompression is simple and *very* fast.
  - Requires no memory for decompression.
  - The decompressors can be squeezed into less than 200 bytes of code.
  - Focuses on compression levels for generating pre-compressed data which achieve a quite competitive compression ratio.
  - Allows you to dial up extra compression at a speed cost in the compressor. The speed of the decompressor is not reduced.
  - Algorithm is thread safe.
  - Algorithm is lossless.
- UCL supports in-place decompression.
- UCL and the UCL algorithms and implementations are distributed under the terms of the GNU General Public License (GPL) { auf Deutsch }. Special licenses for commercial and other applications are available by contacting the author.

Another method of speeding up the kernel load phase of bootup is to use DMA Copy Of Kernel On Startup

How to implement or use

Get UCL from following URL and use sample command "uclpack"

```
http://www.oberhumer.com/opensource/ucl/download/ucl-1.03.tar.gz
```

untar the file, build , and use the sample command "uclpack", located at: ucl-1.03/examples/uclpack in the untar'ed source tree.

Expected Improvement

The case study below is intended to show a performance improvement in decompressing a sample file system and sample kernel.

Resources

- UCL can be obtained at: http://www.oberhumer.com/opensource/ucl/

Projects

- lzop (http://www.lzop.org/ Wikipedia: lzop) uses the miniLZO Lempel-Ziv-Oberhumer (Wikipedia: LZO) algorithm. It has the reputation of extremely fast decompression and a tiny decompressor, but larger compressed files -- apparently faster decompression and better compressed file size than Lempel-Ziv Ross Williams (Wikipedia: LZRW) -- faster than memcopy() on some machines.
- UPX: the the Ultimate Packer for eXecutables (Wikipedia: UPX ; http://upx.sf.net/) uses the UCL algorithm
- gzip (Wikipedia: gzip)
- bzip2 (Wikipedia: bzip2 ; http://www.bzip.org/) has the reputation of giving smaller compressed files and about the same decompression time as gzip (but longer compression times)

[Are there other compressors with better decompression performance than gzip??

Case Studies

Case 1

For this use case, we compiled both uclpack and gzip for the PowerPC platform. Then we ran the programs on the target platform, compressing and decompressing two different file images

- an initrd filesystem image, and a linux kernel image (originally uncompressed).

The size and performance results from running these commands are in the tables below.

Image file:	initrd-2.6.5-1.358		
method	UCL	GZIP	improved %
parameter	-b4194304	-8	.
source file size	819200	819200	.
compressed size	187853	189447	.
compression rate	77.1%	76.9%	0.3%
compression time: user (sec)	5.13	2.03	-152.5%
sys (sec)	0.09	0.06	-36.5%
total (sec)	5.22	2.09	-149.0%
decompression time: user (sec)	0.12	0.3	59.7%
sys (sec)	0.1	0.08	-16.9%
total (sec)	0.22	0.39	43.0%
.	.	.	.
Image file:	vmlinux-2.4.20 for ibm-440gp		
method	UCL	GZIP	improved %
parameter	-b4194304	-8	.
source file size	1810351	1810351	.
compressed size	790250	776807	.
compression rate	56.3%	57.1%	-1.3%
compression time: user (sec)	17.29	6.07	-185.0%
sys (sec)	0.04	0.02	-92.4%
total (sec)	17.33	6.09	-184.6%
decompression time: user (sec)	0.12	0.16	26.1%
sys (sec)	0.03	0.04	35.8%
total (sec)	0.15	0.2	28.2%

Hardware

PPC440GP - 300 MHZ

Kernel Version

Linux kernel running on target was 2.6.11, kernel which was compressed with Linux 2.4.20

Configuration

See above tables for parameters to gzip and ucl

Time without change

[put that here]

Time with change

[put that here]

Case 2

Case 3

Categories:

- Boot Time
- Kernel

> From: eLinux.org

Filesystem Information

Boot Time with various file systems

Noboru Wakabayashi of Hitachi provided the following report.

On the OMAP Innovator he built the following file systems:

- CRAMFS
- CRAMFS with XIP
- ROMFS
- JFFS2
- ext2 over RAM disk

He measured the time to initialize these file systems by logic analyzer. This was done by modifying the busybox init program to make LED turn on red. When innovator power on, the LED lights up green. So the time from the light turning from green to red was measured. Also, he measured the time using KFI (from start_kernel() to to_usrspace()).

Results are shown in the following table. (The result is average of 10 trials for each configuration.)

Configuration/Filesystem	logic analyzer(sec)	KFI(usec)
CRAMFS*	3.638850	842789.1
CRAMFS with XIP	2.788076	764141.3
CRAMFS with XIP and PLPJ	2.583988	551735.5
ROMFS	3.510876	839078.2
JFFS2*	4.822642	1241068.4(log full)
ext2 over RAM disk	cannot measure	2952081.6(log full)

- JFFS2: JFFS2 required much time at first boot time, so he measured from the 2nd starting.
- CRAMFS: At first, he also measured the time with CONFIG_KFI by logic analyzer. The result is 4.324660 sec. It costs more than without CONFIG_KFI. So, he measured file systems without CONFIG_KFI when he used logic analyzer.

The attached zip file has the kfi logfiles for these different tests: no zip found: kfi-results-omap-filesystems.zip

Next he remeasured the time to initialize "CRAMFS with XIP and PLPJ" using the "quiet" option. The result is 280676 usec from start_kernel() to to_userspace(). The above result is 551735.5 usec.The time is reduced about 50%.

The following table shows output from 'kd' on the kfi logfile.

output from 'kd' on the kfi logfile

Function	Count	Time	Average	Local	Max-sub	Ms count
do_basic_setup	1	114068	114068	509	do_initcalls	1
mem_init	1	110376	110376	490	free_all_bootmem_node	1
free_all_bootmem_node	1	109378	109378	12	free_all_bootmem_core	1
free_all_bootmem_core	1	109366	109366	109366	-	0
schedule	10	84482	8448	34	do_schedule	10
do_schedule	10	84448	8444	574	switch_to	9
do_initcalls	1	84159	84159	3831	device_init	1
switch_to	9	83874	9319	83874	-	0
register_proc_table	22	39161	1780	13079	register_proc_table	18
device_register	11	22297	2027	415	device_add	11
device_add	11	21882	1989	1439	kobject_add	11
device_init	1	20633	20633	30	net_dev_init	1
tifb_init	1	18759	18759	844	register_framebuffer	1
register_framebuffer	1	13092	13092	88	take_over_console	1
take_over_console	1	13004	13004	819	redraw_screen	1
kobject_add	15	12996	866	738	create_dir	15
setup_arch	1	12542	12542	131	paging_init	1
paging_init	1	12411	12411	386	free_area_init_node	1
create_dir	15	12258	817	3625	populate_dir	9
free_area_init_node	1	12025	12025	30	free_area_init_core	1
free_area_init_core	1	11995	11995	7496	__alloc_bootmem_node	1
rs_init	1	11794	11794	377	printk	3
inet_init	1	11696	11696	1718	ip_init	1
redraw_screen	2	11247	5623	871	do_update_region	1
printk	18	10870	603	10870	-	0
net_dev_init	1	10334	10334	3102	ethif_probe	1

Probe times for various file systems

As part of work supported by Sony/Matsushita, Todd Poynor got the following numbers using KFI on a 200MHz IBM 405GP "Walnut" evaluation board with a 100MHz core bus and 33MHz PCI bus. A Seagate Barracuda ATA IV 60GB disk drive is cabled to one of the two IDE interfaces on a Promise Ultra66 PCI-IDE bridge card (PDC20262 chipset). All of the drivers for PCI, IDE, PCI-IDE disk, and EXT2 filesystem are built into the kernel.

Boot execution time of IDE/PCI-IDE/MS-DOS partition drivers: 262 msecs. This includes the time to probe and identify the IDE drive device and read the disk partition information from the drive. We booted the kernel with option hdf=none to turn off the slave device on interface ide2, so that it would not be probed. We also modified the kernel to turn off probes of the second IDE interface on the Promise card. (This was prior to fixing the "ide\=noprobe" option bug. If you don't turn off probing the second empty IDE interface then probling takes 1.3 seconds on both a PPC 405GP and a MIPS ITE8172!)

About 250 msecs of the time is spent during the bus probe in repeated calls to ide_delay_50ms() during probe and drive identification, which busywaits (in order to let the IDE controller make progress before polling for status or to allow previous operations to complete). Reading capacity info, etc. also blocks using a wait_for_completion(). The MSDOS partition code also locks pages, which can call schedule() to wait for locks.

If the IDE drivers are made modules for delayed initialization, allowing concurrent module initialization with application execution, then kernel preemption is turned off for about 252 msecs during init of the ide-probe-mod module, which could significantly interfere with real-time response of other threads. (This was verified using the CONFIG_PREEMPT_TIMES feature that gives preemption lock times in /proc/latencytimes, which is also supported in the CELF kernel.) Because the Big Kernel Lock (BKL) is held during module initialization, preemption is disabled while the module init routines runs and uses busywait calls, but preemption is allowed when CPU-yielding wait calls are employed (the linux scheduler drops and reacquires the BKL in this case).

So we changed the ide_delay_50ms() busywaits to call schedule_timeout() instead (this is also in the CELF kernel; select CONFIG_IDE_PREEMPT), resulting in a 9.68 msec maximum preempt off time. Note that if you're not using modules but are instead building the drivers statically into the kernel, then the CPU-yielding calls do add some amount of time to the total execution time due to context-switch overhead, etc.

My coworker Dave Singleton also did some analysis and improvement of IDE on the MIPS ITE8172 (again for Sony/Matsushita). He found that with his 7200RPM Maxtor drive, he could reduce the 50ms probe delays to 1ms with no problem. With this, plus some context switch removal and the other optimizations given above, the following boot times were observed, by filesystem type:

{{{ ext2: 167 milliseconds ext3: 457 milliseconds xfs: 236 milliseconds }}}

He explains: "Both EXT3 and XFS file systems cause a log replay at boot/mount time. To improve this time the log recovery feature was by passed in the case of XFS. The log was not replayed and the root file system was mounted readonly. The first init script after booting remounted the root file system readwrite and replayed the log to ensure file system integrity. No such changes were made to EXT3, which is the reason it had the slowest boot times of all 3 file system types."

Category:

- File Systems

> From: eLinux.org

Hardcode kernel module info

Sony found that with the 2.4 kernel, and an extremely large kernel loadable module, it could take the kernel module loader up to 3 seconds just to scan the relocation table for the module in order to load it.

The following patch allows one to pass some arguments to the kernel command line to hardcode some of the information that is detected at module load time. This reduced the module load time significantly for this situation.

This is a very specialized problem, but in case some one finds this useful, here's a more detailed description of the problem, and a patch:

Detailed problem description

We've found that `module_frob_arch_sections(ppc)` takes a long time to find unique symbols in the relocation sections of the kernel module's elf header.

Our particular kernel module had a relocation section with 13757 entries(2749 unique ones) in it, and for this module, we measured 3 seconds to process this section by count_reloc function.

(There may be a way to not generate all this many entries, or not do the relocations in insmod etc... but we don't know about it at this time.)

We made a patch to just work around this problem by just passing the size info by the argument when insmod happens, and not do actual counting.

-Whether we should think of fixing the count algorithm or other proper way of fixing this. -Whether we should think of an option not to do any relocations at all

Patch for 2.4.17 kernel

Download the patch at: Elf_plt_info.patch

This patch makes it possible to specify the core size and init size, avoiding the call to module_frob_arch_sections()

Categories:

- Boot Time
- Kernel

> From: eLinux.org

IDE No Probe

Preliminary Draft under construction

Contents

Introduction

This page describes the feature "forced IDE noprobe". Normally, the Linux kernel supports the ability to avoid probing for specific IDE drives and IDE interfaces, using a command line argument specified at boot time. The command line argument is "hdx==noprobe" or "idex==noprobe".

Under certain conditions this command line argument is not observed by the Linux kernel (at least for version 2.4.20). This "feature" forces the kernel to observe the "noprobe" command for IDE interfaces.

The fix here records the fact that "noprobe" was specified on the command line (via "the forcenoprobe" field), and skips probing the device if it is set, even if the driver has changed the setting of the noprobe field. This is all a quick hack, and a more systemic rewrite of noprobe handling would be more appropriate for community work, if this is still needed.

Rationale

Probing IDE devices can take a few seconds during system startup. It is especially painful to probe interfaces which are known not to be used in an embedded device.

Downloads

Patch

- Patch for CELF tree is isolated here: ide-noprobe.patch
- ⚠THIS PATCH (IN ITS ISOLATED FORM) HAS NOT BEEN VERIFIED
- Patch for 2.4.xx is *needed*
- Don't know if patch for 2.6.xx is needed

Utility programs

None.

How To Use

- Apply this patch to your CELF tree (or verify that it is already present)
- use "ide1==noprobe" on the kernel command line
  - see the documentation on ide driver command line parameters in the file:

    ```
    drivers/ide/ide.c
    ```

  - The documentation is in a comment right before the routine

    ```
    ide-setup()
    ```

    , at about line 3200 in the file.
- verify that no probing is done for ide1
- measure the time savings from using the patch and the "noprobe" option

Sample Results

Experimental Results #1

Todd Poynor of MontaVista measured the effect of specifying "hdf==none ide3==noprobe" (avoid probing ide2 slave and both ide3 devices) on a 200MHz IBM 405GP "Walnut" evaluation board with a 33MHz PCI bus. A Seagate Barracuda ATA IV 60GB disk drive was cabled to one of the two IDE interfaces on a Promise Ultra66 PCI-IDE bridge card (PDC20262 chipset). All of the drivers for PCI, IDE, PCI-IDE disk, and EXT2 file system were built into the kernel.

The time to init IDE was 1.3 seconds when the missing devices were probed, and about 230 milliseconds when "hdf==none ide3==noprobe" was specified. Thus, the resulting bootup time savings were about 1.1 seconds.

Experimental Results #2

Experimental Results #3

Details

More details on the original problem in case you're interested...

drivers/ide/ide.c: ide_setup() is called early in start_kernel sequence (via parse_options -> checksetup -> __setup() macro mechanism). It calls init_hwif_data to init each interface to "noprobe==1". Then parses cmd line params such as "ide1==noprobe", setting noprobe back to 1.

Later, drivers such as the PCI-IDE bridge driver, at

```
drivers/ide/ide-pci.c: ide_setup_pci_device
```

, resets the value of noprobe according to whether there's a valid interface there:

```
        hwif->noprobe == !hwif->io_ports[IDE_DATA_OFFSET];
```

This undoes the effect of any "noprobe" cmd line option.

Future Work

Here is a list of things that could be worked on for this feature:

- get an exhaustive list of places where noprobe is reset by the kernel
- find out why community overrides this flag in places?
- maybe refactor to be a "forcenoprobe" option?
- find out if the same problem exists in 2.6.xx kernels

Categories:

- Boot Time
- Kernel

> From: eLinux.org

Include modules in kernel image

For best performance do not load modules at init time. Instead include those modules in the kernel image.

This will give a bigger kernel (as the modules are in it) so loading and starting the kernel will be slower. However this time is recovered because there are no modules to be loaded (which is slightly more expensive than having them into the kernel).

TODO: perform measurements to quantify the gain

Category:

- Kernel

> From: eLinux.org

Initcall Debug

Introduction

Passing the option "initcall_debug" on the kernel command line will cause timing information to be printed to the console for each initcall. initcalls are used to initialize statically linked kernel drivers and subsystems and contribute a significant amount of time to the Linux boot process. The output looks like this:

```
calling ipc_init+0x0/0x28 @ 1
msgmni has been set to 42
initcall ipc_init+0x0/0x28 returned 0 after 1872 usecs
```

You can use 'dmesg' to see the messages after the kernel has booted. With a short 'sed' script, you can reorder the final 'timing' line, and sort the initcalls numerically by duration. Here is a command to do this:

```
dmesg -s 128000 | grep "initcall" | sed "s/\(.*\)after\(.*\)/\2 \1/g" | sort -n
```

Sample output

Here is some sample output from the above command sequence. This was on an old X86-based desktop system. Printk Times was turned on (hence the extra timestamp on each line.)

```
24 msecs [ 2.237177] initcall acpi_button_init+0x0/0x51 returned 0
28 msecs [ 0.763503] initcall init_acpi_pm_clocksource+0x0/0x16c
returned 0
32 msecs [ 0.348241] initcall acpi_pci_link_init+0x0/0x43 returned 0

33 msecs [ 0.919004] initcall inet_init+0x0/0x1c7 returned 0
33 msecs [ 5.282722] initcall psmouse_init+0x0/0x5e returned 0
54 msecs [ 2.979825] initcall e100_init_module+0x0/0x4d returned 0
71 msecs [ 0.650325] initcall pnp_system_init+0x0/0xf returned 0
91 msecs [ 0.872402] initcall pcibios_assign_resources+0x0/0x85
returned 0
187 msecs [ 4.369187] initcall ehci_hcd_init+0x0/0x70 returned 0
245 msecs [ 2.777161] initcall serial8250_init+0x0/0x100 returned 0
673 msecs [ 5.098052] initcall uhci_hcd_init+0x0/0xc1 returned 0
830 msecs [ 4.067279] initcall piix_init+0x0/0x27 returned 0
1490 msecs [ 8.290606] initcall ip_auto_config+0x0/0xd70 returned 0
```

Notes

Using initcall_debug increases the amount of messages produced by the kernel during system boot. It's a good idea to increase the printk log buffer size to avoid overflowing the log buffer. To do this, increase the value of CONFIG_LOG_BUF_SHIFT from 14 to 18. This increases the log buffer from the default size of 16k to 256K.

You will need to enable CONFIG_PRINTK_TIME and CONFIG_KALLSYMS in your kernel configuration for this to work correctly. The first option displays printk timings and the second option ensures that function names are printed rather than memory addresses.

Categories:

- Boot Time
- Kernel

> From: eLinux.org

Kernel Function Trace

Contents

Introduction

Kernel Function Trace (KFT) is a kernel function tracing system, which uses the "-finstrument-functions" capability of the gcc compiler to add instrumentation callouts to every function entry and exit. The KFT system provides for capturing these callouts and generating a trace of events, with timing details. KFT is excellent at providing a good timing overview of kernel procedures, allowing you to see where time is spent in functions and sub-routines in the kernel.

The main mode of operation with KFT is to use the system with a dynamic trace configuration. That is, you can set a trace configuration after kernel startup, using the `/proc/kft` interface, and retrieve trace data immediately. However, another (special) mode of operation is available, called STATIC_RUN mode, where the configuration for a KFT run is configured and compiled statically into the kernel. This mode is useful for getting a trace of kernel operation during system bootup (before user space is running).

The KFT configuration lets you specify how to automatically start and stop a trace, whether to include interrupts as part of the trace, and whether to filter the trace data by various criteria (for minimum function duration, only certain listed functions, etc.) KFT trace data is retrieved by reading from `/proc/kft_data` after the trace is complete.

Tools are supplied to convert numeric trace data to kernel symbols, and to process and analyze the data in a KFT trace.

Basic Use

Documentation for KFT is available (as of 2.6.12) in Documentation/kft.txt, after applying the kft-all-in-one.patch.

An online guide is provided at Using Kernel Function Trace

Here's a presentation about KFT usage:

- Presentation: Learning the Kernel and Finding Performance Problems with KFI
- Sample trace used with presentation: omap-serial_init.trace.txt

KFT used to be called KFI (for Kernel Function Instrumentation). For prior releases of KFT, see KernelFunctionInstrumentation

Download

Patches

- Download directory with recent versions: ftp://dslab.lzu.edu.cn/pub/kft/
  - This is a fairly slow link - you can download the patch for 2.6.21 here: kft-all-in-one-2.6.21.patch
- Patches for Linux 2.6.8.1, 2.6.11 and 2.6.12: see the Patch Archive page (available as an all-in-one patch or a tar archive of broken-out patches)
- Patch for Linux 2.6.11: (can just download kfi-2.patch)
- Patch for Linux 2.6.7 (for x86 only): kfi-26-test1.patch
- Patch for CELF kernel (based on linux-2.4.20): kfi-24-test4.patch

KFT utilities

KFT includes several helper scripts which are located in the kernel `scripts` directory:

- addr2sym - convert function addresses to symbols in the trace data
- kd - KFT dump - does filtering, sorting, analysis and trace formatting of KFT trace logs
- mkkftrun.pl - used during building the kernel to convert a configuration file into a C file to be compiled into the kernel
- sym2addr - convert function names to addresses in a KFT configuration file (for a dynamic trace)

See Documentation/kft.txt, in the kernel source tree after applying the patch, for instructions on using these programs.

How To Use

- download both the patch
- apply the patch in the kernel top-level directory:
- patch -p1 \<kft.patch
- read the rest of the instructions in the Documentation/kft.txt file. (my apologies for being lazy!)

Adding platform support for the kft clock source

The current patch (from Sep 2005), uses sched_clock() as the clock source for kft_readclock(). sched_clock() is new in the 2.6 kernel, and returns a 64-bit value containing nanoseconds (not necessarily relative to any particular time base, but assumed to be monotonically increasing, and relatively frequency-stable.)

If your platform has good support for sched_clock(), then KFT should work for you unmodified. If not, you may wish to do one of two things:

- improve support for sched_clock() in your board port, or
- write a custom kft_readclock() routine.

A "good" sched_clock() routine will provide at least microsecond resolution on return values. Some architectures have sched_clock() returning values based on the `jiffy` variable, which on many embedded platforms only has resolution to 10 milliseconds.

There are some sample custom kft_readclock() routines in the current patch for different architectures.

Issues

Here is a list of things that need more work:

- may need to add noinstrument attributes for some time-critical code (need to check this)
  - maybe check "Function Trace in KDB" patch for help with this

Overhead

Mitsubishi measured the overhead of KFI (the predecessor to KFT). The period is from start_kernel() to smp_init().

Platform was: SH7751R 240MHz (Memory Clock 80MHz)

`With KFI : 922.419 msec Without KFI : 666.982 msec Overhead : 27.69%`

Similar technologies

There are other technologies for doing call traces or kernel profiling that are similar to KFT. Some of these are mentioned on the Kernel Instrumentation page. One that is very similar is a kernel trace mechanism for use with KDB. A patch was posted to LKML in January of 2002. See the message: http://www.uwsg.iu.edu/hypermail/linux/kernel/0201.3/0888.html

Filter Q&A

Tim asked the question:

Q. Is there a way to adjust the trigger or filters to reduce the memory usage?

A. The memory usage is determined by the size of the log, which is specified by `logentries` in the KFT configuration. If `logentries` is not specified, it defaults to a rather large number (20,000 in the current code). To use a smaller trace log, specify a smaller number of logentries in the KFT configuration.

The use of triggers and filters can help you fit more data (or more pertinent data) into the log, so you can more readily see the information you are interested in.

By setting start and stop triggers with a narrower "range" of operation, then the amount of data put into the log will be more limited. For example, the default configuration for a static trace uses

`trigger start entry start_kernel trigger stop entry to_userspace`

This will trace EVERYTHING that the kernel does between those two routines. However, you can limit tracing to a much smaller time area of kernel initialization using better triggers. Here is an example showing a triggers for just watching mem_init(): `trigger start entry mem_init trigger stop exit mem_init`

Filters are also vital to reduce the number of entries the trace log. With no time filters in place, KFT will log every single function executed by the kernel. This will quickly overrun the log (no matter what size you have reserved with `logentries`.

When using KFT to find long-duration functions in the kernel, we usually are not interested in routines that execute quickly, and instead use something like "filter mintime 500" to filter out routines taking less than 500 microseconds.

Sample results

Here is an excerpt from a KFI log trace (processed with addr2sym). It shows all functions which lasted longer than 500 microseconds, from when the kernel entered start_kernel() to when it entered to_userspace().

kft log output (excerpt)

`Kernel Instrumentation Run ID 0`

Logging started at 6785045 usec by entry to function start_kernel Logging stopped at 8423650 usec by entry to function to_userspace

Filters: 500 usecs minimum execution time

Filter Counters:

Execution time filter count = 896348 Total entries filtered = 896348 Entries not found = 24

Number of entries after filters = 1757

```
Entry     Delta     PID          Function                     Called At
    1         0       0               start_kernel   L6+0x0
   14      8687       0                setup_arch   start_kernel+0x35
   39       891       0              setup_memory   setup_arch+0x2a8
   53       872       0  register_bootmem_low_pages   setup_memory+0x8f
   54       871       0              free_bootmem   register_bootmem_low_pages+0x95
   54       871       0         free_bootmem_core   free_bootmem+0x34
  930      7432       0               paging_init   setup_arch+0x2af
  935      7427       0           zone_sizes_init   paging_init+0x4e
  935      7427       0            free_area_init   zone_sizes_init+0x83
  935      7427       0       free_area_init_node   free_area_init+0x4b
  935      3759       0       __alloc_bootmem_node   free_area_init_node+0xc5
  935      3759       0       __alloc_bootmem_core   __alloc_bootmem_node+0x43
 4694      3668       0       free_area_init_core   free_area_init_node+0x75
 4817      3535       0          memmap_init_zone   free_area_init_core+0x2bd
 8807    266911       0                 time_init   start_kernel+0xb6
 8807    261404       0             get_cmos_time   time_init+0x1c
270211     5507       0              select_timer   time_init+0x41
270211     5507       0                  init_tsc   select_timer+0x45
270211     5507       0             calibrate_tsc   init_tsc+0x6c
275718     1638       0              console_init   start_kernel+0xbb
275718     1638       0                  con_init   console_init+0x59
275954      733       0        vgacon_save_screen   con_init+0x288
277376     6730       0                  mem_init   start_kernel+0xf8
277376     1691       0          free_all_bootmem   mem_init+0x52
277376     1691       0     free_all_bootmem_core   free_all_bootmem+0x24
284118    25027       0           calibrate_delay   start_kernel+0x10f
293860      770       0                   __delay   calibrate_delay+0x62
293860      770       0                 delay_tsc   __delay+0x26
294951     1534       0                   __delay   calibrate_delay+0x62
294951     1534       0                 delay_tsc   __delay+0x26
297134     1149       0                   __delay   calibrate_delay+0xbe
297134     1149       0                 delay_tsc   __delay+0x26
  .
  .
  .
1638605       0     145             filemap_nopage   do_no_page+0xef
1638605       0     145                __lock_page   filemap_nopage+0x286
1638605       0     145                io_schedule   __lock_page+0x95
1638605       0     145                   schedule   io_schedule+0x24
1638605       0       5                   schedule   worker_thread+0x217
1638605       0       1             to_userspace   init+0xa6
```

The log is attached here: Media:Kfiboot-9.lst A Delta value of 0 usually means the exit from the routine was not seen.

kft log analysis with 'kd'

Below is a `kd` dump of the data from the above log.

For the purpose of finding areas of big time in the kernel, the functions with high "Local" time are important. For example, `delay_tsc()` is called 156 times, resulting in 619 milliseconds of duration. Other time-consuming routines were: `isapnp_isolate()` , `get_cmos_time()` , `default_idle()` .

The top line showing schedule() called 192 times and lasting over 5 seconds, is accounted wrong due to the switch in execution control inside the schedule routine. (The count of 192 calls is correct, but the duration is wrong.)

```
$ ~/work/kft/kft/kd -n 30 kftboot-9.lst
Function                 Count Time     Average  Local
------------------------ ----- -------- -------- --------
schedule                   192  5173790    26946  5173790
do_basic_setup               1  1159270  1159270       14
do_initcalls                 1  1159256  1159256      627
__delay                    156   619322     3970        0
delay_tsc                  156   619322     3970   619322
__const_udelay             146   608427     4167        0
probe_hwif                   8   553972    69246      126
do_probe                    31   553025    17839       68
ide_delay_50ms             103   552588     5364        0
isapnp_init                  1   383138   383138       18
isapnp_isolate               1   383120   383120   311629
ide_init                     1   339778   339778       22
probe_for_hwifs              1   339756   339756      103
ide_scan_pcibus              1   339653   339653       13
init_setup_piix              2   339640   169820        0
ide_scan_pcidev              2   339640   169820        0
piix_init_one                2   339640   169820        0
ide_setup_pci_device         2   339640   169820      242
probe_hwif_init              4   339398    84849       40
time_init                    1   266911   266911        0
get_cmos_time                1   261404   261404   261404
ide_generic_init             1   214614   214614        0
ideprobe_init                1   214614   214614        0
wait_for_completion          6   194573    32428        0
default_idle               183   192589     1052   192589
io_schedule                 18   171313     9517        0
__wait_on_buffer            14   150369    10740      141
i8042_init                   1   137210   137210      295
i8042_port_register          2   135318    67659      301
__serio_register_port        2   135017    67508        0
```

kft nested call trace with 'kd -c'

Below is a `kd -c` trace of the data from a log taken from a PPC440g platform, from a (dynamic) trace of the function do_fork().

Here is the configuration file that was used: `new begin trigger start entry do_fork trigger stop exit do_fork end`

Here is the first part of the trace in nested call format: Times (Entry, Duration and Local) are in micro-seconds. Note the timer interrupt during the routine.

```
      Entry      Duration   Local       Pid    Trace
         4      20428        209       33 do_fork
         7          6          6       33 |  alloc_pidmap
        18       2643         84       33 |  copy_process
        21        114         19       33 |  |  dup_task_struct
        24          8          6       33 |  |  |  prepare_to_copy
        27          2          2       33 |  |  |  |  sub_preempt_count
        35         22          9       33 |  |  |  kmem_cache_alloc
        38          2          2       33 |  |  |  |  __might_sleep
        43         11          9       33 |  |  |  |  cache_alloc_refill
        49          2          2       33 |  |  |  |  |  sub_preempt_count
        60         65          6       33 |  |  |  __get_free_pages
        63         59         14       33 |  |  |  |  __alloc_pages
        65          3          3       33 |  |  |  |  |  __might_sleep
        71          3          3       33 |  |  |  |  |  zone_watermark_ok
        77         37         17       33 |  |  |  |  |  buffered_rmqueue
        80          4          4       33 |  |  |  |  |  |  __rmqueue
        86          3          3       33 |  |  |  |  |  |  sub_preempt_count
        92          3          3       33 |  |  |  |  |  |  bad_range
        98          2          2       33 |  |  |  |  |  |  __mod_page_state
       103          8          5       33 |  |  |  |  |  |  prep_new_page
       106          3          3       33 |  |  |  |  |  |  |  set_page_refs
       117          2          2       33 |  |  |  |  |  zone_statistics
       141         25          4       33 |  |  do_posix_clock_monotonic_gettime
       143         21          6       33 |  |  |  do_posix_clock_monotonic_get
       146         15          6       33 |  |  |  |  do_posix_clock_monotonic_gettime_parts
       149          9          6       33 |  |  |  |  |  getnstimeofday
       152          3          3       33 |  |  |  |  |  |  do_gettimeofday
       169          3          3       33 |  |  copy_semundo
       174         41         17       33 |  |  copy_files
       177         19          9       33 |  |  |  kmem_cache_alloc
       180          2          2       33 |  |  |  |  __might_sleep
       185          8          5       33 |  |  |  |  cache_alloc_refill
       188          3          3       33 |  |  |  |  |  sub_preempt_count
       200          3          3       33 |  |  |  count_open_files
       209          2          2       33 |  |  |  sub_preempt_count
       218         19          8       33 |  |  kmem_cache_alloc
       220          2          2       33 |  |  |  __might_sleep
       225          9          6       33 |  |  |  cache_alloc_refill
       229          3          3       33 |  |  |  |  sub_preempt_count
       241          2          2       33 |  |  sub_preempt_count
       246        216          9       33 |  |  kmem_cache_alloc
       249        199        199       33 |  |  |  __might_sleep
             !!!! start
       253        151         63       33 timer_interrupt
       256          8          6       -1 !  profile_tick
       259          2          2       -1 !  !  profile_hit
       267         61         15       -1 !  update_process_times
       270          8          5       -1 !  !  account_system_time
       273          3          3       -1 !  !  !  update_mem_hiwater
       281          8          5       -1 !  !  run_local_timers
       284          3          3       -1 !  !  !  raise_softirq
       293         27         16       -1 !  !  scheduler_tick
```

. . .

To see the full trace, go to the KftDoForkTrace page.

Categories:

- Boot Time
- Measuring
- Kernel Function Trace

> From: eLinux.org

Kernel Instrumentation

Here is a listing of some instrumentation systems for the kernel:

Contents

Existing Instrumentation Systems

TimePegs

Andrew Morton's system for measuring intervals between kernel events:

See http://www.zipworld.com.au/~akpm/linux/timepeg.txt

Patches at:

http://www.zip.com.au/~akpm/linux/index.html#timepegs

Printk Times

Produces printk's with extra time data on them. As of kernel 2.6.11 this is part of the mainline kernel enabled by CONFIG_PRINTK_TIME. Previous versions can add it via a very simple patch. It works for bootup time measurements, or other places where you can just jam in a printk or two.

See Printk Times

Boot Tracer

Starting from 2.6.28 the kernel has this new feature to optimize the boot time. It records the timings of the initcalls. Its aim is to be parsed by the scripts/bootgraph.pl tool to produce graphics about boot inefficiencies, giving a visual representation of the delays during initcalls. Users need to boot with the "initcall_debug" and "printk.time=1" parameters, and run "dmesg | perl scripts/bootgraph.pl > output.svg" to generate the final data.

Kernel Function Instrumentation (KFI)

A system which uses a compiler flag to instrument most of the functions in the kernel. Timing data is recorded at each function entry and exit. The data can be extracted and displayed later with a command-line program.

The kernel portion of this is available in the CELF tree now.

Grep for CONFIG_KFI.

See the page Kernel Function Instrumentation page for some preliminary notes.

FIXTHIS - need to isolate this as a patch.

Linux Trace Toolkit

See Linux Trace Toolkit

Kernel Tracer (in IKD patch)

This is part of a general kernel tools package, maintained by Andrea Arcangeli.

See http://www.kernel.org/pub/linux/kernel/people/andrea/ikd/README

The ktrace implementation is in the file kernel/debug/profiler.c It was originally written by Ingo Molnar, Richard Henderson and/or Andrea Arcangeli

It uses the compiler flag -pg to add profiling instrumentation to the kernel.

Function trace in KDB

Last year (Jan 2002) Jim Houston sent a patch to the kernel mailing list which provides support compiler-instrumented function calls.

See http://www.ussg.iu.edu/hypermail/linux/kernel/0201.3/0888.html

ftrace

Ftrace is a simple function tracer which initially came from the -rt patches but was mainlined in 2.6.27. Compiler profiling features are used to insert an instrumentation call (with gcc *-pg* option) that can be overwritten with a NOP sequence to ensure overhead is minimal with tracing disabled (this is enabled through CONFIG_DYNAMIC_FTRACE). There are a number of tracers in the kernel that use ftrace to trace high level events such as irq enabling/disabling preemption enabling/disabling, scheduler events and branch profiling.

The interface to access ftrace can be found in */debugfs/tracing*, and is very extensively documented in Documentation/trace/ftrace.txt.

Steven Rostedt, ftrace's main developer, gave a conference on ftrace at the Ottawa Linux Symposium 2008. The video and slides.

SystemTap / Kprobes

SystemTap is a sophisticated kernel instrumentation tool that can be scripted with its own language to gather information about a running kernel. It uses the Kprobes infrastructure to implement it's tracing.

Notes

Some random thoughts on instrumentation:

- Most instrumentation systems need lots of memory to buffer the data produced
- Some instrumentation systems support filters or triggers to allow for better control over the information saved
- instrumentation systems tend to introduce overhead or otherwise interfere with the thing they are measuring
    - instrumentation systems tend to pollute the cache lines for the processor
- There doesn't seem to be a single API to support in-kernel timing instrumentation which is supported on lots of different architectures. This is the main reason for CELF's current project to define an Instrumentation API

Category:

- Kernel

> From: eLinux.org

Kernel XIP

This page describes the use of Kernel Execute-In-Place as a bootup time reduction technique.

Contents

Description

Execute-in-Place (Wikipedia entry) is a method of executing code directly from long-term storage, instead of first loading it into RAM.

When the kernel is executed in place, the bootloader does not have to:

1. read the kernel from flash or
2. decompress the kernel and
3. write the kernel to RAM.

How to implement or use

TODO: describe how to achieve the technique (config options, command args, etc.)

see Kernel XIP Instructions For OMAP

Expected Improvement - about .5 seconds

The expected improvement from using this technique depends on the size of the kernel, and the time to load it and decompress it from persistent storage.

In general, time savings of about .5 seconds have been observed.

Resources

Projects

- Configure Linux For XIP describes experience with using both Kernel XIP and application XIP.
- In this e-mail, David Woodhouse described issues with implementing support for KERNEL XIP in flash. The requirements here are a bit different from supporting KERNEL XIP in ROM, since the flash may be unreadable during certain flash operations. Therefore, portions of the kernel must be copied to RAM, and certain kernel operations must be disallowed when the flash is unavailable.

Specifications

TODO: list or link to CELF specifications related to this technique

Patches

- Kernel 2.6.10 now includes XIP support:

> ARM PATCH 2154/2: XIP kernel for ARM
>
> Patch from Nicolas Pitre
>
> This patch allows for the kernel to be configured for XIP. A lot of people are using semi hacked up XIP patches already so it is a good idea to have a generic and clean implementation supporting all ARM targets. The patch isn't too intrusive.
>
> It involves:
>
> - modifying the kernel entry code to map separate .text and .data sections in the initial page table, as well as relocating .data to ram when needed
> - modifying the linker script to account for the different VMA and LMA for .data, as well as making sure that .init.data gets relocated to ram
> - adding the final kernel mapping with a new MT_ROM mem type
> - distinguishing between XIP and non-XIP for bootmem and memory resource declaration
> - and adding proper target handling to Makefiles.
>
> While at it, this also cleans up the kernel boot code a bit so the kernel can now be compiled for any address in ram, removing the need for a relation between kernel address and start of ram. Also throws in some more comments.
>
> And finally the _text, _etext, _end and similar variables are now declared extern void instead of extern char, or even extern int. That allows for operations on their address directly without any cast, and trying to reference them by mistake would yield an error which is a good thing.
>
> Tested both configurations: XIP and non XIP, the later producing a kernel for execution from ram just as before.
>
> Signed-off-by: Nicolas Pitre Signed-off-by: Russell King

Case Studies

Case 1 - XIP on Arctic III PowerPC board

XIP was used on a PowerPC board, with the following results:

- Hardware: PowerPC 405LP Arctic III, running at 266 MHZ
- Kernel Version: MontaVista Linux CEE 3.0 (based on 2.4.20)
- Configuration: Features built statically into the kernel included: Arctic ethernet, audio, and MTD; 405LP LCD and touchscreen; 405 onchip I2C; and pinned TLBs; Dynamic Power Management; preemptible kernel with selected

spinlock breaking; serial driver and serial console (kernel messages are disabled for boot time measurements); TCP/IP (IP addresses are configured after boot) with network devices, network packet filtering, packet protocol, and IP multicast; virtual terminal; UNIX domain sockets and UNIX98 PTYs; Linux Driver Model; and /proc, sysfs, tmpfs, ramfs, cramfs, devpts filesystems.
- Time without change: 1357 milliseconds
- Time with change: 894 milliseconds
- Total Reduction in boot time: 463 milliseconds

Table of bootup times:

Boot Stage	**Non-XIP Time**	**XIP Time**
Copy kernel to RAM	85 ms	12 ms *
Decompress kernel	453 ms	0 ms
Kernel time to initialize (time to first user space program)	819 ms	882 ms
Total kernel boot time	1357 ms	894 ms
Reduction:	*	**463 ms**

- still have to copy data segment

Thanks to Todd Poynor of MontaVista for providing this information.

Case 2 - XIP on OMAP Innovator

XIP was used on a TI OMAP (Innovator board), with the following results:

- Hardware: TI OMAP 1510, running at 168 MHZ
- Kernel Version: 2.4.20 (precursor to CELF tree)
- Configuration: [need to put config information here]
- see Kernel XIP Instructions For OMAP

Boot Stage	**Non-XIP Time Kernel compressed**	**Non-XIP Time Kernel not compressed**	**XIP Time**
Copy kernel to RAM	56 ms	120 ms	0 ms
Decompress kernel	545 ms	0 ms	0 ms
Kernel time to initialize (time to first user space program)	88 ms	208 ms	110 ms
Total kernel boot time	689 ms	208 ms	110 ms
Reduction:	*	**481 ms**	**579 ms**

Thanks to Hiroyuki Machida of Sony for providing this information.

Case 3 - comparing NOR XIP with OneNAND quick-copy to RAM

- Hardware: TI OMAP 5912, running at 196 MHZ (OSK5912 from Spectrum Digital)
- Kernel Version: 2.6.10-omap1 (binary size is about 2MBytes uncompressed)

Dongjun Shin of Samsung Electronics reports:

As I've mentioned in AG meeting, we've done some boot time measurements on OMAP 5912 target platform (OSK5912 from Spectrum Digital). We've done this experiment in order to identify the timing gap between NOR XIP and NAND shadowing. Here is the result (the number represents time in microseconds).

The column noted as "XIP tuning" means that we changed the NOR I/F setting of OMAP (EMIFS) so that the synchronous read is used instead of (default) asynchronous read.

In case of OneNAND, only 1Kbytes of initial part of OneNAND can be used as XIP region and we used 1Kbytes IPL for loading u-boot. Shadowing means that kernel copy (to RAM) is used.

The reason why the kernel initialization time are broken into 2 phases is that we used timer register for measurement and the timer is initialized during kernel booting. You can just add the values for 2 phases to get the total kernel booting time.

Boot stage	NOR		OneNAND	
	XIP		Shadowing	
	Normal	Tuning	Compressed	Uncompressed
Boot loader CPU frequency	96MHz		96 MHz	
Boot loader (IPL)	0	0	5,999	5,999
Boot loader (u-boot)	388,146	372,538	356,821	356,810
Copy kernel to RAM	0	0	35,029	56,884
Decompress kernel	0	0	1,178,481	0
Kernel time to initialize - 1 phase	18,964	12,826	9,091	9,119
Kernel time to initialize - 2 phase	61,176	51,263	50,118	50,126
Total	468,287	436,626	1,635,540	478,938
times are in microseconds				

- Related info
  - omap patch archive
  - Samsung NAND flash memory datasheet
  - OneNAND e-brochure

Questions

TimRiker asks:

- What is the ram/rom footprint of these?
- Are we close to using sram only for some implementations?
- Has anyone looked at romfs and XIP user space?

Implementation Notes (from the field)

- Discussion about XIP when flash might be in use - note mention of '__xipram' attribute (for partial XIP??)

Notes on configuring Linux for XIP (for PPC)

- Notes on configuring XIP

Using XIP with U-Boot on Arm

Wolfgang Denks, the primary author of the UBoot bootloader, wrote the following:

```
>> Yes. But... _Does_ mkimage -x put header on the front of it?

Yes, it does.

>>> > * You program the resulting image at 0x10004000.
>>> >
>>> > What is programmed at 0x10004000 ?  The xipImage code or the uboot header?
>
>>
>> The u-boot headers, yes. Thats wrong. But how to use mkimage -x then?
>> Is the header-caused offset known?

Yes. The U-Boot header is 64 bytes.

U-Boot expects (and verifies) that the entry point is  equal  to  the load address plus the size of the U-Boot header
.
```

Lots more details are in the thread (split across months in the archives):

- Thread: Nov. 2004
- http://lists.arm.linux.org.uk/pipermail/linux-arm-kernel/2004-December/025674.html Thread: Dec. 2004]

How to determine offsets for sections

Dick Johnson talks about how to set the physical address for ELF sections by editing the kernel link files.

```
On Fri, 21 Oct 2005, Sreeni wrote:
>> Hi,<br>
>><br>
>> I have a montavista XIP kernel running on ARM and my kernel will be in<br>
>> the flash. Since its XIP, I know that the ".text" portion of the<br>
>> kernel will be executed from flash but that ".data" needs to be placed<br>
>> in SDRAM. Now my question is - based on what offset this data will be<br>
>> placed?<br>
>><br>
>> My SDRAM physicall address starts at 3000_0000 and flash starts at<br>
>> 0100_0000. when i allocated a global variable in the kernel module and<br>
>> when i try to check its actually physical address using virt_to_phys,<br>
>> its giving me the address in the range of 0100_0000 ~ 0600_0000 which<br>
>> is my flash (the PAGE_OFFSET doesn't work in case of XIP).<br>
>><br>
>> Can you please help in knowing the physical address of my .data<br>
>> portion in this situation.<br>
>><br>
>> Thanks<br>
>> Shree<br>
</code><br>
I don't know about the ARM in particular, but if you look in ../arch/arm/boot/compressed/vmlinux.lds.in,
you will see that this linker-file simply allocates the start addresses of each section as the next
available address. The same is true of ../arch/arm/boot/bootp.lds. If you expect to have code the data
elements and stack accessed at a specific physical offset, you modify the linker files().

Note that "." means "right here", just like '$' in many assemblers. You can specify a physical offset
simply as:

 ENTRY(_start)
 SECTIONS
 {
   . == 0x01000000  <==== like this for code
   .text : {
    ...
    ... }
    .rodata : { }
    . == 0x30000000 <==== like this data
    .data : {  }
    .bss  : {  }
 }

In the above, we have put .rodata (initialized ASCII stuff) right after the code in the .text section.
You may need to extract this from the binary blob to put into your NVRAM.

Also, any initialzed data needs to be relocated to your writable SDRAM and the .bss stuff needs to be
zeroed. This is non-trivial. You may want to create a ".reloc" section which contains your initialized
data, put it in your flash, and relocate it at startup.

...

Cheers,
Dick Johnson
```

Categories:

- Boot Time
- Bootloader
- Kernel XIP
- System Size

> From: eLinux.org

Optimize RC Scripts

Contents

RC Scripts Speed-Up

This material was excerpted from a document with the following copyright statement:

It was submitted as input to the forum by Sony Corporation, on April 8, 2004.

Introduction

The init scripts of the existing Embedded Linux distribution are the shell scripts to be executed with 'bash'. To reduce system boot time, some modifications to be applied to the scripts and faster shell interpreter to be used. The document describes the BusyBox optimizations; the init scripts modifications; how to reduce the system boot time using the optimized BusyBox, and to optimize shell scripts for BusyBox. Also, the benchmark procedure and optimization results are described.

Purpose of Feature

The init scripts execution time, i.e. the time interval between the start of the init process and the start of user applications, must be reduced.

Feature requirements

- The modified init scripts must be run with "bash" as well as the BusyBox "ash" shell.
- The execution time of the init scripts and the total system boot time must be reduced.
- The guide to speed-optimization of the init scripts must be provided.

Acceptance Criteria

- The modified init scripts are able to successfully run with "bash" as well as the BusyBox "ash" shell.
- The execution time of the init scripts with BusyBox is reduced in the comparison with the original init scripts with "bash"; the total system boot time is not greater then 5 sec.
- The guide to speed-optimization of the init scripts is available.

BusyBox Optimization

Since ’bash’ and GNU utilities are very heavy applications, BusyBox is useful to reduce system boot time. BusyBox combines tiny versions of many common UNIX utilities (shellutils, fileutils, etc.) into a single small executable. The commands and utilities included in BusyBox are divided into the classes: built-ins and applets. The built-ins are simply invoked as functions, and applets are invoked by means of the ’fork/exec’ system calls. Also, BusyBox scripts can use external commands and utilities.

Usage of the BusyBox built-ins is rather than the applets and external commands by performance reasons, because the ’fork/exec’ system calls are very heavy and they make the main contribution to shell inefficiency. Because the original BusyBox is only size-optimized, the following features must be considered from the performance standpoint:

- Each command including built-ins within pipes are forked.
- Each back-quoted command is forked.
- The ’echo’, ’test’, and ’[’ commands and other most frequent commands in the scripts are implemented as applets.

To avoid such drawbacks, we optimized BusyBox 1.00-pre3 in order to speed up script execution. The BusyBox optimizations for the ’ash’ shell are listed below:

- The set of shell commands and utilities is implemented as built-ins.
- The invoked ’cat’ command at the beginning of pipes is eliminated and file descriptors are passed only into the next command of the pipe.

The following shell commands and utilities are implemented as built-ins in the optimized BusyBox ’ash’ shell: ’,’, ’:’,’[’, ’alias’, ’break’, ’cd’, ’chdir’, ’continue’, ’echo’, ’eval’, ’exec’, ’exit’, ’export’, ’false’, ’kill’, ’let’, ’local’, ’pidof’, ’pwd’, ’read’, ’readonly’, ’return’, ’set’, ’shift’, ’test’, ’times’, ’trap’, ’true’, ’type’, ’ulimit’, ’umask’, ’unalias’, ’unset’, ’wait’.

NOTES:

- This need seems to be addressed for all applets at least in Busybox 1.1.1 (and perhaps in earlier versions). The "Shells -> Standalone shell" configuration setting is supposed to address this need. See the menuconfig help for details.
- In newer BusyBox releases (1.13.0 and maybe even earlier releases) the most frequently used applets, 'test', 'echo' can be configured as being 'built-in'. The newer BusyBox releases are also smaller in size and can save some extra milliseconds in execution of startup scripts.

Known Problems

The following BusyBox commands work in the different manner from the ’bash’ commands and GNU utilities: ’nice’, ’find’, ’mount’, ’umount’, ’init’, ’halt’, ’shutdown’, ’syslogd’, ’klogd’, ’hwclock’, ’cron’, ’anacron’, ’crontab’, 'pidof'.

If the different behaviour is unwanted or these BusyBox applets do not provide the necessary utility, use the external commands instead of them, or simply do not configure them as applets.

How to Optimize Init Scripts for BusyBox

Follow the rules listed below to reduce execution time for the init scripts:

- Do not use unnecessary codes in the scripts.

- Replace external commands and utilities with the BusyBox built-ins as far as possible.
- Do not use the piped commands as far as possible.
- Reduce the number of commands within a pipe.
- Do not use the back-quoted commands as far as possible.

The main goal of such optimization is to reduce the number of the "fork/exec" calls during a script execution.

Examples of Init Scripts Optimization

The following examples demonstrate how the recommendations of shell scripts optimization can be applied to the init scripts.

Unnecessary codes elimination

This example demonstrates the elimination of duplicate codes from the "mountswap.sh" and "checkrootfs.sh" scripts: the command "swapon" runs one time in the modified scripts.

- Before optimization:

checkrootfs.sh:

```
if [ "$FSCKSWAP" !== no ]
then
   if [ -x /sbin/swapon ]
   then
       mount -n /proc
       if ! grep -qs resync /proc/mdstat
       then
           [ "$VERBOSE" !== no ] && echo "Activating swap..."

           swapon -a 2> /dev/null
       fi
       umount -n /proc
    fi
fi
```

mountswap.sh:

```
grep -qs resync /proc/mdstat || swapon -a 2> /dev/null
```

- After optimization:

checkrootfs.sh:

mountswap.sh:

```
if [ "$FSCKSWAP" !== "no" ]; then
    if [ ! ‘grep -qs resync /proc/mdstat‘ ]
    then
        log_status_msg "Starting $DESC: " -n
        log_status_msg "$BASENAME1" -n
        $DAEMON1 $ARGS1
        RET==$?
        if [ $RET -eq 0 ]; then
            log_success_msg ". " -n
        else
            log_failure_msg " failed ($RET: $ERROR)."
            return 1
        fi
    fi
fi
```

Built-in usage

This example demonstrates usage of the "echo" built-in instead of the external command "printf" in the "nfs-common" script.

- Before optimization:

nfs-common:

```
   printf "Starting $DESC:"
   printf " statd"
```

- After optimization: init-functions:

```
log_status_msg() {
   if [ "$1" !== "" ] && [ "$1" !== "-n" ]
   then

      if [ "$2" == "-n" ]
      then echo -n "$1"
      else echo "$1"
      fi

    fi
return 0
}
```

nfs-common.sh:

```
 log_status_msg "Starting $DESC: " -n
 log_status_msg "$BASENAME1" ?n
```

Piped command usage

This example demonstrates the elimination of the piped commands (the example is hypothetic, because the init scripts do not contain such inefficiencies).

- Before optimization:

```
cat /proc/mounts | grep ext3 | cut -d’ ’ -f2,3
```

- After optimization:

```
sed -n ’s/^[^ ]* \([^ ]*\) \(ext3\) .*$/\1 \2/p’ /proc/mounts
```

This example demonstrates the reduction of the commands in the pipe (the example is also hypothetic). Note, the optimized version does not invoke the "fork" call, because the "cat" optimization is used.

- Before optimization:

```
cat /etc/passwd | grep user | wc -l | tr -d ’’ | sed ’s/ *//’
```

- After optimization:

```
cat /etc/passwd | grep -c user
```

Back-quoted command usage

This example demonstrates the back-quoted command elimination (the example is hypothetic, because the init scripts do not contain such inefficiencies).

- Before optimization:

```
if [ "‘grep rpcuser /etc/passwd‘" !== "" ]
then
    echo "rpcuser"
else
    echo "no rpcuser"
fi
```

- After optimization:

```
if grep rpcuser /etc/passwd >/dev/null
then
    echo "rpcuser"
else
    echo "no rpcuser"
fi
```

Init Scripts Optimization

The existing init scripts were modified to reduce their execution time following the recommendations, which are described above.

Benchmark Environment and Procedure

To estimate the results of the init script optimization and the BusyBox usage, the TI OMAP 1510 Innovator platform is used. To measure the duration of the kernel loading, the KFI support is used (to measure the init script execution time, the KFI support is disabled). The measurements are performed on the systems with/without XIP support To take measurements without the XIP support, the following kernel configuration is used:

```
CONFIG_ARM==y
CONFIG_UID16==y
CONFIG_RWSEM_GENERIC_SPINLOCK==y
CONFIG_EXPERIMENTAL==y
CONFIG_ADVANCED_OPTIONS==y
CONFIG_MODULES==y
CONFIG_KMOD==y
CONFIG_ARCH_OMAP==y
CONFIG_OMAP_INNOVATOR==y
CONFIG_INNOVATOR_MISSED_IRQS==y
CONFIG_ARCH_OMAP1510==y
CONFIG_CLOCK_COUNTS_DOWN==y
CONFIG_CPU_32==y
CONFIG_CPU_ARM925T==y
CONFIG_CPU_ARM925_CPU_IDLE==y
CONFIG_CPU_ARM925_I_CACHE_ON==y
CONFIG_CPU_ARM925_NON_STREAMING_ON==y
CONFIG_CPU_ARM925_D_CACHE_ON==y
CONFIG_CPU_32v4==y
CONFIG_KERNEL_START==0xc0000000
CONFIG_ZBOOT_ROM_TEXT==0
CONFIG_ZBOOT_ROM_BSS==0
CONFIG_NET==y
CONFIG_SYSVIPC==y
CONFIG_SYSCTL==y

CONFIG_MAX_USER_RT_PRIO==100
CONFIG_MAX_RT_PRIO==0
CONFIG_FPE_NWFPE==m
CONFIG_KCORE_ELF==y
```

```
CONFIG_BINFMT_AOUT==m
CONFIG_BINFMT_ELF==y
CONFIG_OMAP1510_PM==y
CONFIG_DPM==y
CONFIG_BOOT_FREQ==y
CONFIG_OMAP_ARM_168MHZ==y
CONFIG_OMAP1510_DPM==y
CONFIG_INNOVATOR_DPM==y
CONFIG_CMDLINE=="mem==32M console==ttyS0,115200n8 noinitrd root==/dev/null rootflags==physaddr==0x0260000"
CONFIG_ALIGNMENT_TRAP==y
CONFIG_PREEMPT==y
CONFIG_LOCK_BREAK==y
CONFIG_MTD==y
CONFIG_MTD_PARTITIONS==y
CONFIG_MTD_CONCAT==y
CONFIG_MTD_CHAR==y
CONFIG_MTD_BLOCK==y
CONFIG_MTD_CFI==y
CONFIG_MTD_GEN_PROBE==y
CONFIG_MTD_CFI_ADV_OPTIONS==y
CONFIG_MTD_CFI_NOSWAP==y
CONFIG_MTD_CFI_GEOMETRY==y
CONFIG_MTD_CFI_B2==y
CONFIG_MTD_CFI_I1==y
CONFIG_MTD_CFI_INTELEXT==y
CONFIG_MTD_CFI_AMDSTD==y
CONFIG_MTD_OMAP==y
CONFIG_MTD_OMAP_0==y
CONFIG_MTD_OMAP_1==y
CONFIG_BLK_DEV_LOOP==m
CONFIG_BLK_DEV_RAM==y
CONFIG_BLK_DEV_RAM_SIZE==4096
CONFIG_BLK_DEV_INITRD==y
CONFIG_PACKET==m
CONFIG_NETFILTER==y
CONFIG_UNIX==y
CONFIG_INET==y
CONFIG_IP_MULTICAST==y
CONFIG_IP_PNP==y
CONFIG_NETDEVICES==y
CONFIG_NET_ETHERNET==y
CONFIG_NET_VENDOR_SMC==y
CONFIG_SMC9194==y
CONFIG_PPP==m
CONFIG_PPP_MULTILINK==y
CONFIG_PPP_ASYNC==m
CONFIG_PPP_DEFLATE==m
CONFIG_PPPOE==m
CONFIG_IRDA==m
CONFIG_IRLAN==m
CONFIG_IRNET==m
CONFIG_IRCOMM==m
CONFIG_OMAP_SIR==m
CONFIG_INPUT==m
CONFIG_INPUT_KEYBDEV==m
CONFIG_INPUT_MOUSEDEV==m
CONFIG_INPUT_MOUSEDEV_SCREEN_X==240
CONFIG_INPUT_MOUSEDEV_SCREEN_Y==320
CONFIG_INPUT_EVDEV==m
CONFIG_VT==y
CONFIG_VT_CONSOLE==y
CONFIG_SERIAL==y
CONFIG_SERIAL_CONSOLE==y
CONFIG_UNIX98_PTYS==y

CONFIG_UNIX98_PTY_COUNT==256
CONFIG_I2C==m
CONFIG_I2C_ALGOBIT==m
CONFIG_I2C_OMAP1510==m
CONFIG_I2C_CHARDEV==m
CONFIG_I2C_PROC==m
CONFIG_SENSORS==y
```

```
CONFIG_SENSORS_OTHER==y
CONFIG_SENSORS_EEPROM==m
CONFIG_WATCHDOG==y
CONFIG_OMAP_WATCHDOG==m
CONFIG_OMAP_RTC==m
CONFIG_RV5C387_RTC==m
CONFIG_RV5C387_RTC==m
CONFIG_VIDEO_DEV==m
CONFIG_VIDEO_PROC_FS==y
CONFIG_AUTOFS4_FS==m
CONFIG_EXT3_FS==m
CONFIG_JBD==m
CONFIG_FAT_FS==m
CONFIG_MSDOS_FS==m
CONFIG_VFAT_FS==m
CONFIG_JFFS_FS==m
CONFIG_JFFS_FS_VERBOSE==0
CONFIG_JFFS2_FS==y
CONFIG_JFFS2_FS_DEBUG==0
CONFIG_CRAMFS==y
CONFIG_CRAMFS_LINEAR==y
CONFIG_CRAMFS_LINEAR_XIP==y
CONFIG_ROOT_CRAMFS_LINEAR==y
CONFIG_TMPFS==y
CONFIG_RAMFS==y
CONFIG_PROC_FS==y
CONFIG_DEVPTS_FS==y
CONFIG_EXT2_FS==m
CONFIG_NFS_FS==y
CONFIG_NFS_V3==y
CONFIG_NFSD==m
CONFIG_NFSD_V3==y
CONFIG_SUNRPC==y
CONFIG_LOCKD==y
CONFIG_LOCKD_V4==y
CONFIG_SMB_FS==m
CONFIG_MSDOS_PARTITION==y
CONFIG_SMB_NLS==y
CONFIG_NLS==y
CONFIG_NLS_DEFAULT=="iso8859-1"
CONFIG_NLS_CODEPAGE_437==m
CONFIG_PC_KEYMAP==y
CONFIG_FB==y
CONFIG_DUMMY_CONSOLE==y
CONFIG_FB_OMAP==y
CONFIG_FBCON_ADVANCED==y
CONFIG_FBCON_CFB16==y
CONFIG_FBCON_FONTWIDTH8_ONLY==y
CONFIG_FBCON_FONTS==y
CONFIG_FONT_8x8==y
CONFIG_FONT_ACORN_8x8==y
CONFIG_SOUND==m
CONFIG_SOUND_OMAP==m
CONFIG_SOUND_OMAP_AIC23==m
CONFIG_INNOVATOR_TS==y
CONFIG_MMC==m
CONFIG_OMAP_MMC==m
CONFIG_INSTANT_ON==y
CONFIG_DEFAULT_LPJ==414720
CONFIG_INSTANT_ON_LPJ==414720
CONFIG_FRAME_POINTER==y

CONFIG_DEBUG_INFO==y
CONFIG_DEBUG_KERNEL==y
CONFIG_MAGIC_SYSRQ==y
CONFIG_DEBUG_BUGVERBOSE==y
CONFIG_DEBUG_ERRORS==y
CONFIG_ZLIB_INFLATE==y
CONFIG_ZLIB_DEFLATE==y
```

To take measurements with the XIP support, the following kernel configuration is used:

```
CONFIG_ARM==y
CONFIG_UID16==y
CONFIG_RWSEM_GENERIC_SPINLOCK==y
CONFIG_EXPERIMENTAL==y
CONFIG_ADVANCED_OPTIONS==y
CONFIG_MODULES==y
CONFIG_KMOD==y
CONFIG_ARCH_OMAP==y
CONFIG_OMAP_INNOVATOR==y
CONFIG_INNOVATOR_MISSED_IRQS==y
CONFIG_ARCH_OMAP1510==y
CONFIG_CLOCK_COUNTS_DOWN==y
CONFIG_CPU_32==y
CONFIG_CPU_ARM925T==y
CONFIG_CPU_ARM925_CPU_IDLE==y
CONFIG_CPU_ARM925_I_CACHE_ON==y
CONFIG_CPU_ARM925_NON_STREAMING_ON==y
CONFIG_CPU_ARM925_D_CACHE_ON==y
CONFIG_CPU_32v4==y
CONFIG_KERNEL_START==0xc0000000
CONFIG_ZBOOT_ROM_TEXT==0
CONFIG_ZBOOT_ROM_BSS==0
CONFIG_NET==y
CONFIG_SYSVIPC==y
CONFIG_SYSCTL==y
CONFIG_MAX_USER_RT_PRIO==100
CONFIG_MAX_RT_PRIO==0
CONFIG_XIP_ROM==y
CONFIG_XIP_PHYS_ADDR==60400
CONFIG_FPE_NWFPE==m
CONFIG_KCORE_ELF==y
CONFIG_BINFMT_AOUT==m
CONFIG_BINFMT_ELF==y
CONFIG_OMAP1510_PM==y
CONFIG_DPM==y
CONFIG_BOOT_FREQ==y
CONFIG_OMAP_ARM_168MHZ==y
CONFIG_OMAP1510_DPM==y
CONFIG_INNOVATOR_DPM==y
CONFIG_CMDLINE=="mem==32M console==ttyS0,115200n8 noinitrd root==/dev/null rootflags==physaddr==0x0260000"
CONFIG_ALIGNMENT_TRAP==y
CONFIG_PREEMPT==y
CONFIG_LOCK_BREAK==y
CONFIG_MTD==y
CONFIG_MTD_DEBUG==y
CONFIG_MTD_DEBUG_VERBOSE==0
CONFIG_MTD_PARTITIONS==y
CONFIG_MTD_CONCAT==y
CONFIG_MTD_CHAR==y
CONFIG_MTD_BLOCK==y
CONFIG_MTD_CFI==y
CONFIG_MTD_GEN_PROBE==y
CONFIG_MTD_CFI_ADV_OPTIONS==y
CONFIG_MTD_CFI_NOSWAP==y
CONFIG_MTD_CFI_GEOMETRY==y
CONFIG_MTD_CFI_B2==y
CONFIG_MTD_CFI_I1==y

CONFIG_MTD_CFI_INTELEXT==y
CONFIG_MTD_CFI_AMDSTD==y
CONFIG_MTD_OMAP==y
CONFIG_MTD_OMAP_1==y
CONFIG_BLK_DEV_LOOP==m
CONFIG_BLK_DEV_RAM==y
CONFIG_BLK_DEV_RAM_SIZE==4096
CONFIG_BLK_DEV_INITRD==y
CONFIG_PACKET==m
CONFIG_NETFILTER==y
CONFIG_UNIX==y
CONFIG_INET==y
CONFIG_IP_MULTICAST==y
```

```
CONFIG_IP_PNP==y
CONFIG_NETDEVICES==y
CONFIG_NET_ETHERNET==y
CONFIG_NET_VENDOR_SMC==y
CONFIG_SMC9194==y
CONFIG_PPP==m
CONFIG_PPP_MULTILINK==y
CONFIG_PPP_ASYNC==m
CONFIG_PPP_DEFLATE==m
CONFIG_PPPOE==m
CONFIG_IRDA==m
CONFIG_IRLAN==m
CONFIG_IRNET==m
CONFIG_IRCOMM==m
CONFIG_OMAP_SIR==m
CONFIG_INPUT==m
CONFIG_INPUT_KEYBDEV==m
CONFIG_INPUT_MOUSEDEV==m
CONFIG_INPUT_MOUSEDEV_SCREEN_X==240
CONFIG_INPUT_MOUSEDEV_SCREEN_Y==320
CONFIG_INPUT_EVDEV==m
CONFIG_VT==y
CONFIG_VT_CONSOLE==y
CONFIG_SERIAL==y
CONFIG_SERIAL_CONSOLE==y
CONFIG_UNIX98_PTYS==y
CONFIG_UNIX98_PTY_COUNT==256
CONFIG_I2C==m
CONFIG_I2C_ALGOBIT==m
CONFIG_I2C_OMAP1510==m
CONFIG_I2C_CHARDEV==m
CONFIG_I2C_PROC==m
CONFIG_SENSORS==y
CONFIG_SENSORS_OTHER==y
CONFIG_SENSORS_EEPROM==m
CONFIG_WATCHDOG==y
CONFIG_OMAP_WATCHDOG==m
CONFIG_OMAP_RTC==m
CONFIG_RV5C387_RTC==m
CONFIG_RV5C387_RTC==m
CONFIG_VIDEO_DEV==m
CONFIG_VIDEO_PROC_FS==y
CONFIG_AUTOFS4_FS==m
CONFIG_EXT3_FS==m
CONFIG_JBD==m
CONFIG_FAT_FS==m
CONFIG_MSDOS_FS==m
CONFIG_VFAT_FS==m
CONFIG_JFFS_FS==m
CONFIG_JFFS_FS_VERBOSE==0
CONFIG_JFFS2_FS==y
CONFIG_JFFS2_FS_DEBUG==0
CONFIG_CRAMFS==y
CONFIG_CRAMFS_LINEAR==y
CONFIG_CRAMFS_LINEAR_XIP==y

CONFIG_ROOT_CRAMFS_LINEAR==y
CONFIG_TMPFS==y
CONFIG_RAMFS==y
CONFIG_PROC_FS==y
CONFIG_DEVPTS_FS==y
CONFIG_EXT2_FS==m
CONFIG_NFS_FS==y
CONFIG_NFS_V3==y
CONFIG_NFSD==m
CONFIG_NFSD_V3==y
CONFIG_SUNRPC==y
CONFIG_LOCKD==y
CONFIG_LOCKD_V4==y
CONFIG_SMB_FS==m
CONFIG_MSDOS_PARTITION==y
CONFIG_SMB_NLS==y
```

```
CONFIG_NLS==y
CONFIG_NLS_DEFAULT=="iso8859-1"
CONFIG_NLS_CODEPAGE_437==m
CONFIG_PC_KEYMAP==y
CONFIG_FB==y
CONFIG_DUMMY_CONSOLE==y
CONFIG_FB_OMAP==y
CONFIG_FBCON_ADVANCED==y
CONFIG_FBCON_CFB16==y
CONFIG_FBCON_FONTWIDTH8_ONLY==y
CONFIG_FBCON_FONTS==y
CONFIG_FONT_8x8==y
CONFIG_FONT_ACORN_8x8==y
CONFIG_SOUND==m
CONFIG_SOUND_OMAP==m
CONFIG_SOUND_OMAP_AIC23==m
CONFIG_INNOVATOR_TS==y
CONFIG_MMC==m
CONFIG_OMAP_MMC==m
CONFIG_TRACE==y
CONFIG_TRACE_BOOT==y
CONFIG_INSTANT_ON==y
CONFIG_DEFAULT_LPJ==414720
CONFIG_INSTANT_ON_LPJ==414720
CONFIG_FRAME_POINTER==y
CONFIG_DEBUG_INFO==y
CONFIG_DEBUG_KERNEL==y
CONFIG_MAGIC_SYSRQ==y
CONFIG_DEBUG_BUGVERBOSE==y
CONFIG_DEBUG_ERRORS==y
CONFIG_ZLIB_INFLATE==y
CONFIG_ZLIB_DEFLATE==y
```

The set of the init scripts of the consumer packages is divided into the minimal and optional subsets. The following scripts belong to the minimal subset: ’bootmisc.sh’, ’checkfs.sh’, ’checkroot.sh’, ’hwclock.sh’, ’modutils.sh’, ’mountall.sh’, ’networking.sh’, ’urandom.sh’.

The following scripts belong to the optional subset: ’anacron.sh’, ’cron.sh’, ’devfsd.sh’, ’devshm.sh’, ’ifupdown.sh’, ’rmnologin.sh’, ’syslog.sh’.

Thus,

- the minimal packages are: "initscripts", "util-linux", "modutils", and "netbase";
- the optional packages are: "initscripts", "anacron", "cron", "devfsd", "ifupdown", "sysklogd", "util-linux", "modutils", and "netbase".

The init scripts are used with the "bash", BusyBox, 0.60.3 and the optimized BusyBox 1.00-pre3. The sizes of the shell executables are:

Since profiling tools slow up program execution, the Linux Trace Toolkit (LTT) is not used to measure the execution time of the init scripts.

Shell	**Size**
bash	562 Kb
BusyBox 0.60.3	872 Kb
BusyBox 1.00-pre3	210 Kb
The optimized BusyBox 1.00-pre3	259 Kb

To obtain the start time of the init scripts, the modified "init" utility is used. The following patch for the "init" utility is applied:

```
diff -Naur sysvinit-2.78.orig/src/init.c sysvinit-2.78/src/init.c
--- sysvinit-2.78.orig/src/init.c 2000-02-11 14:17:02.000000000 +0300
+++ sysvinit-2.78/src/init.c 2003-12-08 20:20:15.000000000 +0300
@@ -684,7 +684,7 @@
    (void) tcgetattr(fd, &tty);
    tty.c_cflag &== CBAUD|CBAUDEX|CSIZE|CSTOPB|PARENB|PARODD;
-   tty.c_cflag |== HUPCL|CLOCAL;
+   tty.c_cflag |== CREAD|HUPCL|CLOCAL;
    tty.c_cc[VINTR] == 3; /* ctrl('c') */
    tty.c_cc[VQUIT] == 28; /* ctrl('\\') */
@@ -2443,6 +2443,13 @@
    FILE *fp;
    char *p;
    int f, fd;
+
+#ifdef TIME_MEASURE
+ struct timeval current;
+ if (!gettimeofday (&current, 0))
+ printf ("INIT: started at %ld\n", (current.tv_sec * 1000) + (current.tv_usec / 1000));
+#endif
 /* Get my own name */
 if ((p == strrchr(argv[0], '/')) !== NULL)
```

To obtain the time of the end of the init script execution, the following program is used in the inittab:

```
#include <sys/time.h>
#include <stdio.h>
int main (int argc, char **argv)
{
    struct timeval current;
    if (!gettimeofday (&current, 0))
    printf ("printtime: current time %ld ms\n", (current.tv_sec * 1000) + (current.tv_usec / 1000));
    return 0;
}
```

Thus, we consider the sum of the init script execution time and the kernel loading time, which are measured in ways described above, is the total system boot time.

Optimization Results

Measurement results with the XIP support Kernel loading time: 0.6 sec. Init script execution time:

Script set	Shell	Time	%
minimal	bash	3.6 sec.	35%
minimal	BusyBox 0.60.3	5.3 sec.	52%
minimal	optimized BusyBox 1.00-pre3	2.6 sec.	25%
minimal + optional	bash	7.6 sec.	74%
minimal + optional	BusyBox 0.60.3	10.1 sec.	100%
minimal + optional	optimized BusyBox 1.00-pre3	4.9 sec.	48%

Total system boot time (kernel loading time + init script execution time):

Script set	Shell	Time	%
minimal	bash	4.3 sec.	39%
minimal	BusyBox 0.60.3	6.0 sec.	55%
minimal	optimized BusyBox 1.00-pre3	3.2 sec.	29%
minimal + optional	bash	8.2 sec.	76%
minimal + optional	BusyBox 0.60.3	10.7 sec.	100%
minimal + optional	optimized BusyBox 1.00-pre3	5.5 sec.	51%

Measurement results without the XIP support Kernel loading time: 0.4 sec. Init script execution time:

Script set	Shell	Time	%
minimal	bash	4.4 sec.	52%
minimal	BusyBox 0.60.3	5.0 sec.	59%
minimal	optimized BusyBox 1.00-pre3	2.9 sec.	34%
minimal + optional	bash	6.8 sec.	82%
minimal + optional	BusyBox 0.60.3	8.3 sec.	100%
minimal + optional	optimized BusyBox 1.00-pre3	4.6 sec.	55%

*Total system boot time (kernel loading time + init script execution time):

Script set	Shell	Time	%
minimal	bash	4.8 sec.	55%
minimal	BusyBox 0.60.3	5.4 sec.	61%
minimal	optimized BusyBox 1.00-pre3	3.3 sec.	38%
minimal + optional	bash	7.3 sec.	83%
minimal + optional	BusyBox 0.60.3	8.7 sec.	100%
minimal + optional	optimized BusyBox 1.00-pre3	5.0 sec.	57%

Downloads

- busybox-1.00-pre3-optimized.patch
- init_scripts.tgz

Categories:

- Boot Time
- Optimize RC Scripts

> From: eLinux.org

Parallel RC Scripts

Contents

Description

One way to reduce bootup time is to run RC scripts in parallel. RC scripts are normally run in sequence in a desktop configuration of Linux. By running the scripts in parallel, it is possible to take advantage of the multi-processing capabilities of the OS (such as overlapping execution with I/O, etc.)

How to implement or use

See the projects listed below for details on different methods of doing this.

Expected Improvement

[Not determined yet.]

Resources

Projects

- InitNG: a new replacement for SysV init. Boots your system much faster by running as much as possible asynchronously. See InitNG
- IBM article on on using Makefile techniques to express dependencies between services and support parallel service start. See BootFaster
- Richard Gooch project to rewrite boot script system from scratch. Eliminates lots of BSD and SYS V-isms, and introduces dependencies. See boot scripts
- Serel project - for parallelizing service startup. Commands are inserted into RC scripts to cause needed services to start (based on XML database of dependencies). See fastboot

Specifications

- LSB specification for comments in RC Scripts which allow parallization. See [1]

Patches

None.

Case Studies

[None yet.]

Case 1

[put information about an actual use of this technique here. A case study should include:]

Hardware:: [hardware description here] Kernel Version:: [kernel version here] Configuration:: [information about the configuration used here] Time without change:: [put that here] Time with change:: [put that here]

[Add any additional notes as you see fit.]

Case 2

Case 3

Category:

- HOWTOs

> From: eLinux.org

Pre Linking

Contents

Description

Pre-Linking is a mechanism for linking programs to shared libraries ahead of time. In general, every time an application is run it must have its external symbols resolved - looked up in the shared library symbol table, and fixed up in the program binary to refer to the correct offsets in the library. To use prelinking, a special utility is run which does this resolution and fixup once for the program. This saves the cost of linking at runtime.

There is an existing package from RedHat which provides this feature.

A drawback of this is that if the shared library is changed, the fixups are no longer correct, and the program must be fixed-up again. This is much less of an issue in an embedded situation, where the programs and libraries are less likely to change than in a desktop or server Linux system.

Overview of linking

There is an excellent paper with an overview of dynamic linking issues at: Pre Linking Overview This paper describes not only pre-linking, but lazy linking and more exotic systems, like compile-on-load.

Expected Improvement

[This is not measured yet.]

We expect that with use of prelinking, there will be a slight reduction in boot time for Linux system, in the area of initial application loading.

We need to use this system and measure the effect of prelinking for a determined set of applications.

Resources

RedHat prelinking system

- The prelink package is at: http://people.redhat.com/jakub/prelink/
- A white paper is at: prelink

prelink currently supports the following architectures: alpha, arm, cris, i386, ia64, ppc32, ppc64, s390, sh, sparc32, sparc64, x86_64. At present the glibc dynamic linker is required to prelink executables and load prelinked code, uClibc does not support it.

Instructions for using prelinking with Gentoo

The following page has information on how to use prelinking with a Gentoo system:

http://www.gentoo.org/doc/en/prelink-howto.xml

Related Projects

- Prebinding (RelCache) - RelCache (aka ELF prebinding) news http://mail-index.netbsd.org/tech-userlevel/2002/12/04/0017.html
- RelCache vs. Red Hat prelink

http://mail-index.netbsd.org/tech-userlevel/2002/12/01/0000.html

- Resident - Resident Good (comparisons with prebind)

http://www.shiningsilence.com/dbsdlog/2004/01/20/215.html

Specifications

None so far.

Patches

No kernel patches required for kernels 2.4.10 and later.

Case Studies

Case 1 - Panasonic mobile phone prelink

Panasonic used pre-linking on their Linux-based mobile phones. These used a 2.4.x Linux kernel, for an ARM processor. Measuring the time to load a single multimedia application with regular dynamic linking and pre-linking, showed that pre-linking could save a lot of time.

Hardware
ARM9 (unspecified CPU frequency)

Kernel Version
2.4.20 (based on Monta Vista Linux CEE 3.1), glibc 2.3

Time without change
2479 ms

Time with change
125 ms

Source

page 19 of Making Mobile Phone with CE Linux

Case 2

Case 3

Future Work

This item is a work-in-progress, and we are just getting started.

Material from CELF presentations

ARM Prelink

- Japan Jamboree #3
  - http://tree.celinuxforum.org/CelfPubWiki/JapanTechnicalJamboree3#head-1515fb2d64cd91370e9cb2f6ad4847483e729cf3 In the presentation of "Making Mobile Phone with CE Linux", the evaluation of Prelink on ARM architecture was mentioned.
    - by Mr. Mizuyama (Panasonic Mobile)

MIPS Prelink

- Japan Jamboree #13
  - http://tree.celinuxforum.org/CelfPubWiki/JapanTechnicalJamboree13#head-ab59e6354d343ec0a804b5f440d35b5dcc27304c
    - Evaluation report by Mr. Yagi (Mitsubishi)

Categories:

- HOWTOs
- Boot Time

> From: eLinux.org

Preset LPJ

Contents

Introduction

"Preset LPJ" is a feature to avoid the cost associated with calibrating loops_per_jiffy at each boot time.

The value of loops_per_jiffy (LPJ) is normally calculated in the routine calibrate_delay() early in the initialization sequence of the Linux kernel. The cost of doing this operation is independent of the CPU frequency and is about 250 milliseconds on most hardware. The value of LPJ should be the same for the same hardware operating at the same clock frequency. Thus LPJ can be calculated once and used for subsequent boots, and the cost to do the delay calibration can be avoided.

Basically, the patch allows you to specify a preset value for loops_per_jiffy at kernel compile time, or on the command line at kernel boot time.

LKML Discussion

The CELF patch was submitted to LKML on July 10, and was discussed here

Rationale

This saves about 250 milliseconds on a 2.4-based Linux system. The duration of the calibration does not depend on the speed of the processor, but on the value of HZ for a particular architecture, and the number of iterations required to perform the calibration. For a 2.6 version of Linux, HZ is now defined as 1000 for the i386 platform (meaning a HZ duration is 1 millisecond rather than 10 milliseconds as it was for most architectures in the 2.4 version of the Linux kernel). Thus, for i386, the savings is now only about 25 milliseconds. However, many architectures still use a HZ value of 100, so for these architectures this change is still important.

Specification

The forum has written a specification for this feature. It is available at: [Calibrate Delay Avoidance Specification R2]. The specification page has more details about the operation of calibrate_delay() and the need for this feature.

Downloads

⚠ A version of this feature is now included in official (Linus' tree) Linux version 2.6.9-rc2.

- Note that this version does away with the kernel configuration option, and only allows preseting lpj from the kernel command line.

Patch

- No patch, no configuration settings needed for kernel's later than 2.6.9-rc2.
- Patch for 2.6.7 is on the Patch Archive page
- Patch for CELF version 040304 is here: attachment:preset-lpj-1.patch

How To Use

Boot your kernel

- Measure boot time
- In kernel boot messages, read loops_per_jiffy value measured by your kernel. Example: Calibrating delay loop... 187.59 BogoMIPS (lpj==937984)
- If you see no such message in the console, view the /proc/kmsg file or boot your kernel with the "loglevel==8" parameter.

For kernels older than 2.6.9-rc2

- Apply this patch to your CELF kernel
- Configure your kernel with the feature turned on:

```
**Turn on "Fast boot options"
**Turn on "Use preset loops_per_jiffy"
```

- Recompile your kernel

Reboot your kernel

- Provide a preset value for loops_per_jiffy at the kernel command line using the string "lpj==\", where \ is replaced with the correct value for loops_per_jiffy for your platform.
- Measure the new boot time and compare with the original one
- Notice the new message: Calibrating delay loop (skipped)... 187.59 BogoMIPS preset

Sample Results

case 1*2.6.7 with patch

Tim Bird (of Sony) measured the result of using preset lpj on his x86 desktop system. It saved 268 milliseconds of bootup time.

Details:

- Kernel version: CELF Linux kernel (2.4.20-based)
- CPU: Pentium 4 running at 3 GHz

case 2

Richard Griffiths (of Intel) measured the result of using preset lpj on an x86 system. It saved about the same (~268 milliseconds) of bootup time.

Details:

- Kernel version: CELF Linux kernel (2.4.20-based)
- CPU: Celeron running at 1 GHz

case 3

Noboru Wakabayashi (of Hitachi) measured the result of using preset lpj on a TI OMAP (ARM-based) system. It saved about 212 milliseconds.

- Kernel version: CELF Linux kernel (2.4.20-based)
- CPU: OMAP 1510 running at 168 MHz

case 4

Tim Bird measured use of preset-lpj on an x86 desktop system with the 2.6.6 kernel. It saved 25 milliseconds.

Details:

- Kernel version: Linux (kernel.org) 2.6.6 with preset-lpj patch applied
- CPU: Pentium 4 running at 3 GHz

case 5*2.6.8-rc1-mm1

With the new patch, Tim Bird got the following results:

- Kernel version: Linux (kernel.org) 2.6.8-rc1-mm1
- CPU: Pentium 4 running at 3 GHz
- With HZ==1000:

```
**normal boot: calibrate_delay() took 23 milliseconds
**specifying lpj==xxx: calibrate_delay() took 43 microseconds.
```

- With HZ==100:

```
**normal boot: calibrate_delay() took 264 milliseconds
**specifying lpj==xxx: calibrate_delay() took 43 microseconds.
```

case 6

Jyunji Kondo (of Fujitsu Prime Software Technologies) measured the result of using preset lpj on FR-V processor. It saved about 205 milliseconds.

Details:

- Kernel version: 2.6.6
- CPU: FR-V FR450 core running at 360 MHz

Please also refer to the graphic chart in DMA Copy Of Kernel On Startup.

Future Work

Here is a list of things that need to be done with this patch:

- possibly provide lpj validation feature, mentioned by specification

Testing

Testing performed for 2.6.7*for preset-lpj-5.patch (after LKML discussion)

- see if patch applies cleanly to 2.6.7
  - OK
- see if patch applies cleanly to 2.6.7-mm7
  - (preset-lpj-04.06.25.patch applied cleanly, but with offsets for all hunks)
- see if patch applies cleanly to 2.6.7-bk20
  - (preset-lpj-04.06.25.patch applied cleanly, but with offsets for all hunks)
- see if build succeeds on 2.6.7
  - OK
- runtime tests:
  - run with PRESET_LPJ set to 0 (default)
    - result should be: (everything normal)
    - should print lpj value that is used
  - run with PRESET_LPJ set to 0, with lpj==xxx command line
    - should skip calibration, but print BogoMips anyway
    - should NOT?? print lpj value that can be used??
  - run with PRESET_LPJ option set to yyy in config
    - should skip calibration
    - should NOT print lpj value that can be used
  - run with PRESET_LPJ option set to yyy in config, with lpj==xxx command line
    - should skip calibration, and use xxx rather than yyy
    - should NOT print lpj value that can be used
  - run with PRESET_LPJ option set to yyy in config, with lpj==0 command line
    - should perform calibration
    - should print lpj value that is used

Categories:

- Boot Time
- Kernel

> From: eLinux.org

Printk Times

Contents

Introduction

"Printk-times" is a simple technology which adds some code to the standard kernel printk routine, to output timing data with each message. While crude, this can be used to get an overview of the areas of kernel initialization which take a relatively long time. This feature is used by the Bootup Time Working Group to identify areas of the Linux kernel requiring work to improve bootup time, and to measure the improvements of changes made by the working group.

The technology for this feature consists of a patch and a utility program. The patch alters the printk code in the kernel to emit the timing data.

UPDATE: The patch was incorporated into the mainline kernel as of version 2.6.11! Both the feature, and the utility program are now part of mainline Linux!!

With printk-times turned on, the system emits the timing data as a floating point number of seconds (to microsecond resolution) for the time at which the printk started. The utility program shows the time between calls, or it can show the times relative to a specific message. This makes it easier to see the timing for specific segments of kernel code.

Rationale

There are other instrumentation systems for the kernel that have more advanced features than this. However, this system is very simple and robust. It does not require extra programs, interfaces in proc of sysfs, or even a root filesystem, in order to obtain measurements of bootup time. Its weakness is that you can only see timing information for areas of the kernel which have printks. In order to get more detail for an area of interest, you have to add additional printks to the kernel, and re-compile it. Also, printks themselves may add too much timing overhead to the kernel, particularly when the output is over a serial line (which is very common in embedded configurations).

For a system that provides much more detailed timing information, you may want to use Kernel Function Instrumentation instead.

Downloads

Patches

- Patch for x86 for 2.4.20 kernel: instrumented_printk.patch
- Patches (arch-neutral) for 2.6.x kernels are on the Patch Archive page.
- Patch for 2.6.11-rc4: printk-times-3.patch

[Image:Alert.gif] - patch is no longer needed. Printk-times was incorporated into the mainline Linux kernel as of version 2.6.11.

Utility program

The program `show_delta` reads the information from printk output, and displays time values as delta's between printks. This helps to more easily find periods of long time between printks. (This is easier than "eye-balling" the dump, looking for jumps in the time.)

The `show_delta` is in the `/scripts` directory of the kernel source, for kernel versions 2.6.12 and above.

How To Use (2.6 version of kernel)

- If using a kernel version prior to 2.6.11, apply the appropriate patch to your kernel
- When booting the kernel, add the option "printk.time=1" (or "time" for older kernels) on the kernel command line
- You should see extra data at the beginning of each printk line
- Examine the data while it is displayed, or after booting use the "dmesg" command line.
- (See the "trouble-shooting" section below if you only see some, not all, of the bootup messages)
- Collect the kernel printk data with `dmesg`
- save the results to a file, like so: `dmesg >/tmp/bootup_printks`
- alternatively, if you are using a serial console, you can capture the kernel printk output to the capture buffer of your terminal program and save it to a file for use later.
- (Optionally) To see the time spent between successive printks, use the `show_delta` command.
  - `scripts/show_delta /tmp/bootup_printks`
- (Optionally) To avoid the overhead of writing out the kernel messages during bootup, try booting the kernel with the "quiet" kernel command line option.
- If you use a serial console, it is possible for the speed of the serial connection to interfere with the accuracy of the timing data. In this case you may wish to boot the kernel with the "quiet" option, which suppresses printk output during boot. When this option is used, the kernel messages are not printed during bootup, but are still available after booting with the `dmesg` command.
- (Optionally) If you want to configure the kernel to always boot with timing information turned on, and/or you want to see timing information for the kernel BEFORE it parses the kernel command line:
  - Configure your kernel with "Show timing information on printks" turned on.
  - This option is on the the "Kernel hacking" menu of the kernel configuration program.
  - Compile and boot your kernel

runtime control of printk times

You can enable and disable printk timestamps at runtime, by writing to /sys/module/printk/parameters/time.

```
# cat /sys/module/printk/parameters/time
N
# echo 1 >/sys/module/printk/parameters/time
# cat /sys/module/printk/parameters/time
Y
# echo "sample log message" >/dev/kmsg
# dmesg | tail
....
[3814526.197336] sample log message
```

History

Jan Engelhardt supplied a patch against 2.6.17 which allows for runtime control of the printk-times control flag. Here is what he wrote:

> Currently, enabling/disabling printk timestamps is only possible through reboot (bootparam) or recompile. I normally do not run with timestamps (since syslog handles that in a good manner), but for measuring small kernel delays (e.g. irq probing - see parport thread) I needed subsecond precision, but then again, just for some minutes rather than all kernel messages to come. The following patch adds a module_param() with which the timestamps can be en-/disabled in a live system through /sys/module/printk/parameters/printk_time.

The patch was applied in 2.6.18.

Trouble-shooting

- Not all kernel messages are displayed by dmesg
  - The printk-times feature adds a number of bytes at the beginning of each printk message. The default kernel message buffer size may not be sufficient to hold all the messages with this additional overhead. You can increase the kernel message buffer size when compiling the kernel, by adjusting the "Kernel Log buffer size" (found on the "General Setup" menu). Notethat you must also specify a larger buffer read size with "dmesg", with the '-s' option.
    - ex: `dmesg -s 128000 >/tmp/bootup_printks`
- Resolution of timings is very bad.
  - Printk-time uses the routine sched_clock() in the kernel. On some platforms, sched_clock() only has a resolution of 1 jiffy (which may be 10 milliseconds or longer). This means that you will only see the time increment at this resolution, giving imprecise results for printk-times. To correct this problem, the best solution is to implement a good sched_clock() routine for your platform. Sched_clock() returns a 64-bit value which is nanoseconds since some event (usually either since machine power-on, or since time_init() was called.) Many embedded processors have a clock or timer on the System-On-Chip which can provide a good resolution clock source for sched_clock(). It is best if the clock can provide resolution better than 1 microsecond. Note that this only requires a clock running at 1 MHz to achieve this resolution.
- Machine hangs when printk-times is compiled ON.
  - You may experience a panic, hang-up, or some other problem with printk-times turned on. I have seen problems caused by calls to sched_clock() too early in boot sequence. Some platforms don't support calling sched_clock() before memory, interrupts or other architecture-specific items are initialized. (The 'i386' arch is OK, since it's sched_clock() by default uses the TSC, which does not need any prior setup.) In case you have problems on your platform, you may need to adjust the sched_clock() function to return 0 until it is safe to begin operation (usually until after time_init() completes.)

Customizing the printk times clock

Somewhere between 2.6.11 and 2.6.14, the printk routine was modified to use printk_clock() instead of sched_clock() directly. This means that you can override printk_clock() on your platform to use a custom timestamp source different from the one used by sched_clock() (Or, you can avoid calling sched_clock() until it is ready - say, after time_init()).

How To Use (2.4 version of kernel)

- Apply the 2.4-based patch to your kernel
- You MUST set the value for fixed_cpu_khz, in the file `include/asm-386/timex.h` (in the inline macro highres_timer_ticks_to_timeval) to the correct value for your machine.
- to find out the correct value for this, run your system (without the patch or with the option turned off), and 'cat /proc/cpuinfo' Use the value from the "cpu MHZ" line, multiplied by 1000 (that is, with the decimal point removed.)
- Configure your kernel with "Configure timing instrumentation in printk" turned on.
- (This should appear at the bottom of the "General Setup menu of the kernel configuration program.)
- Compile and boot your kernel
- You should see extra data at the beginning of each printk line
- Collect the kernel printk data with `dmesg` save the results to a file, like so: `dmesg >bootup_printks`
- alternatively, if you are using a serial console, you can capture the kernel printk output to the capture buffer of your terminal program and save it to a file for use later.
- (Optionally) Use show_delta to display the time spent between successive printks
- (Optionally) Boot with the "quiet" kernel command line option.
- If you are using a serial console, it is possible that the speed of the serial connection will interfere with the accuracy of the timing data. In this case you may wish to boot the kernel with the "quiet" option, which will suppress printk output during boot. When this option is used, the kernel messages are still be available after booting with the dmesg command.

Sample Results

Below is a sample of kernel messages produced by a 2.6.11 kernel, with "time" specified on the kernel command line. Note that the messages don't start having the time prefix until the kernel command line is processed. Then, the timestamps don't have real values until the timer code in the kernel is initialized. This is fairly typical for non-x86 kernels.

```
Linux version 2.6.11 (tbird@timdesk.am.sony.com) (gcc version 3.4.3) #35 Wed Oct 8 12:09:15 PDT 2008
memsize environment variable not set: assuming 32MB
splash screen environment variable not set: assuming not enabled
CPU revision is: 0001906c
Start xilleon_setup...
Enable memory access in PCI space for DMA
Xilleon GPIO4_SEL: 0x00000200
Xilleon GPIO5_SEL: 0x00000080
Determined physical RAM map:
 memory: 00080000 @ 00000000 (ROM data)
 memory: 00180000 @ 00080000 (reserved)
 memory: 02000000 @ 00200000 (usable)
On node 0 totalpages: 8704
  DMA zone: 8704 pages, LIFO batch:2
  Normal zone: 0 pages, LIFO batch:1
  HighMem zone: 0 pages, LIFO batch:1
Built 1 zonelists
Kernel command line: root=/dev/nfs nfsroot=192.168.2.1:/nfsroot_gtx rw ip=192.168.2.93 time
[    0.000000] Primary instruction cache 32kB, physically tagged, 4-way, linesize 16 bytes.
[    0.000000] Primary data cache 32kB, 4-way, linesize 16 bytes.
[    0.000000] Synthesized TLB refill handler (20 instructions).
[    0.000000] Synthesized TLB load handler fastpath (32 instructions).
[    0.000000] Synthesized TLB store handler fastpath (32 instructions).
[    0.000000] Synthesized TLB modify handler fastpath (31 instructions).
[    0.000000] PID hash table entries: 256 (order: 8, 4096 bytes)
[    0.000000] calculating r4koff... #### mips_hpt_frequency: 246095550
[    0.000000] 00258d1b(2460955) with HZ(100)
[    0.000000] CPU frequency 492.19 MHz
[    0.098499] Using 246.096 MHz high precision timer.
[    0.128948] Dentry cache hash table entries: 8192 (order: 3, 32768 bytes)
[    0.136327] Inode-cache hash table entries: 4096 (order: 2, 16384 bytes)
[    0.145717] Memory: 29608k/32768k available (2147k kernel code, 3120k reserved, 400k data, 124k init, 0k highmem)
[    0. 156719] Calibrating delay loop... 491.52 BogoMIPS (lpj=2457600)
[    0.384932] Mount-cache hash table entries: 512 (order: 0, 4096 bytes)
[    0.392079] Checking for 'wait' instruction...  available.
[    0.399982] NET: Registered protocol family 16
[    0.406901] SCSI subsystem initialized
```

```
[    0.411487] usbcore: registered new driver usbfs
[    0.416508] usbcore: registered new driver hub
[    0.422558] HBIU Device 0000:00:14.0 0 MEM 20000000:27ffffff -> 08000000:0fffffff
[    0.430497] PCU  Device 0000:00:14.0 3 MEM 1c000000:1fffffff -> 1c000000:1fffffff
[    0.438416] PCI  Device 0000:00:14.0 2 MEM 19000000:191fffff -> 10000000:101fffff
[    0.446330] PCI  Device 0000:00:14.2 0 MEM 19200000:19200fff -> 10200000:10200fff
[    0.454241] PCI  Device 0000:00:14.0 1 I/O 00002000:000020ff -> 00002000:000020ff
[    0.466247] uirt: Xilleon IR TV Remote Driver [fmt:12-bit Sony nuirt:1 demod=n nscan=3 debug=n]
[    0.475923] Serial: 8250/16550 driver $Revision: 1.90 $ 4 ports, IRQ sharing disabled
[    0.484350] ttyS0 at MMIO 0x0 (irq = 100) is a 16550A
[    0.489832] ttyS1 at MMIO 0x0 (irq = 99) is a 16550A
[    0.495228] ttyS2 at MMIO 0x0 (irq = 122) is a 16550A
[    0.500699] ttyS3 at MMIO 0x0 (irq = 127) is a 16550A
[    0.506385] io scheduler noop registered
[    0.510541] io scheduler deadline registered
[    0.516049] loop: loaded (max 8 devices)
[    0.523834] GMAC_ETH: Init completed-GMAC
[    0.528498] MUSB Core: Initializing MUSB Driver [npci=1][gadget=no][otg=no][dma=no]
[    0.536585] Probing direct bus [direct=1]
[    0.541245] MGC_LinuxInitController: MUSBxDRC at 0xbe001400, IRQ 130
[    0.547971] MGC_LinuxInitController: Driver instance data at 0x810d2000
[    0.702474] MGC_O_HDRC_CONFIGDATA register contents: 0xde
[    0.708164] MGC_HdrcInit: ConfigData=de (UTMI-8, dyn FIFOs, bulk combine, bulk split, HB-ISO Rx, HB-ISO  Tx, SoftC
onn )
[    0.719350] MGC_LinuxInitController: End 00: FIFO TxSize=0040/RxSize=0040
[    0.726504] MGC_LinuxInitController: End 01: Shared FIFO TxSize=0400/RxSize=0400
[    0.734302] MGC_LinuxInitController: End 02: Shared FIFO TxSize=0400/RxSize=0400
[    0.742096] MGC_LinuxInitController: End 03: FIFO TxSize=0800/RxSize=0000
[    0.749247] MGC_LinuxInitController: End 04: FIFO TxSize=0000/RxSize=0800
[    0.756394] MGC_LinuxInitController: End 05: Shared FIFO TxSize=0100/RxSize=0100
[    0.764188] MGC_LinuxInitController: End 06: Shared FIFO TxSize=0100/RxSize=0100
[    0.771985] MGC_LinuxInitController: End 07: Shared FIFO TxSize=0100/RxSize=0100
[    0.779783] MGC_LinuxInitController: End 08: Shared FIFO TxSize=0100/RxSize=0100
[    0.787576] MGC_LinuxInitController: End 09: Shared FIFO TxSize=0100/RxSize=0100
[    0.795386] MGC_LinuxInitController: New bus @0x804d2e80
[    0.801568] musb-hcd usb0: new USB bus registered, assigned bus number 1
[    0.809014] hub 1-0:1.0: USB hub found
[    0.812976] hub 1-0:1.0: 1 port detected
[    0.916507] ohci_hcd: 2004 Nov 08 USB 1.1 'Open' Host Controller (OHCI) Driver (PCI)
[    0.916679] ohci_hcd 0000:00:14.2: OHCI Host Controller
[    0.922216] ohci_hcd 0000:00:14.2: irq 2, pci mem 0x10200000
[    0.928346] ohci_hcd 0000:00:14.2: new USB bus registered, assigned bus number 2
[    0.948236] hub 2-0:1.0: USB hub found
[    0.952246] hub 2-0:1.0: 2 ports detected
[    0.975448] USB Universal Host Controller Interface driver v2.2
[    0.981849] Initializing USB Mass Storage driver...
[    0.987126] usbcore: registered new driver usb-storage
[    0.992536] USB Mass Storage support registered.
[    0.997407] i2c /dev entries driver
[    1.001451] Misc PCA16 driver minor=62
[    1.005490] NET: Registered protocol family 2
[    1.010225] IP: routing cache hash table of 512 buckets, 4Kbytes
[    1.016822] TCP established hash table entries: 2048 (order: 2, 16384 bytes)
[    1.024317] TCP bind hash table entries: 2048 (order: 1, 8192 bytes)
[    1.031076] TCP: Hash tables configured (established 2048 bind 2048)
[    1.037833] NET: Registered protocol family 1
[    1.536645] GMAC_ETH: Reset called
[    1.540426] GMAC_ETH: Half duplex
[    2.530746] IP-Config: Guessing netmask 255.255.255.0
[    2.536368] IP-Config: Complete:
[    2.539580]       device=eth0, addr=192.168.2.93, mask=255.255.255.0, gw=255.255.255.255,
[    2.548143]      host=192.168.2.93, domain=, nis-domain=(none),
[    2.554384]      bootserver=255.255.255.255, rootserver=192.168.2.1, rootpath=
[    2.562567] Looking up port of RPC 100003/2 on 192.168.2.1
[    2.570535] Looking up port of RPC 100005/1 on 192.168.2.1
[    2.632542] VFS: Mounted root (nfs filesystem).
[    2.637464] Freeing unused kernel memory: 124k freed
```

show_delta results

Sample Results for `show_delta` :

- Printk Times Sample1
  - basic output from show_delta with every line relative to the line previous
- Printk Times Sample2
  - shows output with all lines relative to first line (good for getting the total kernel bootup time)
- Printk Times Sample3
  - shows output relative to a single line in the middle (good for measuring the total time for multi-line items)

Sample Results for dmesg output for 2.6.11-rc4:

- Printk Times Sample4
  - showing truncation of dmesg output

Related Work

Standardized boot instrumentation points

Here are some ideas for additional work needed on this system:

- should add printks to key areas of the kernel (e.g. before each sub-system init), to establish reference timings for various sub-systems
  - this would be useful to find boot time regressions

initcall_debug

A system already exists for finding the amount of time it takes to load all the individual non-core drivers in the kernel. It is called initcall_debug.

See Initcall_Debug

Categories:

- Boot Time
- Measuring
- Printk

From: eLinux.org

Ramdisks demasked

Contents

Ramdisks demasked

Introduction

When a system does not boot as quickly as expected, people sometimes revert to using a populated ram disk. The idea is that a ram disk resides in RAM, and hence is faster than a flash file system (which is correct). However, in the end no performance gain occurs. This page described why this happens and why in general a ram disk is not a good idea.

How ram disks are used

The standard way to use a populated ram disk is to make a ram disk image, load it to memory using the boot loader (e.g. using a grub initrd line in menu.lst) and tell the kernel to use the ram disk (by means of the initrd= and root=/dev/ram0 kernel parameter).

Why is there no performance gain?

This is simple. The ramdisk need to be loaded from flash memory. This takes time. After that there is indeed a speedup because you do not need to access the flash any more. However, the gain from this speedup hardly ever recoups the time needed to load the flash disk in the first place.

Reason for this is that if there is no ram disk only those parts of an executable that are really needed are read from the background memory. Unit of transfer is a page (typically 4K or 8K). So if you were using only a few functions from e.g. glibc only those pages are loaded, not the full glibc (which might be around 1MB). However in case of a ram disk, the whole glibc needs to be loaded. It is easy to see that loading a whole file (as done in the ram disk case) takes more time than reading the pages that you need (as is the case in the non ramk disk case).

Now you might reason that pages might be read multiple times, which contribues to the gain. Technically you are right. Rereading the same page several times will cause some saving (although it still can be argued if this is enough to recover the ram disk load time). However lets look a little bit deeper: If a page is reread it has been dropped from the buffer cache. This means that the kernel is running out of buffer space. If that is the problem then a ram disk is not going to be the solution! Better use no ram disk and give that memory to the kernel, who will most likely be able to use it in a more efficient manner.

But is there no situation when a ram disk will give a gain?

Of course there are. I can think of the following situations:

- If most or all pages of your ram disk are used, especially if your boot loader is faster reading the ram disk than the kernel is (this could be related to compression and/or to less read overhead).
- If the source of the ram disk might disappear (e.g. if the ram file system resides on the network or on a removable disk that may be removed).
- If execution start time is more important than boot time (then it helps if glibc and friends are in RAM). (this might also be the case if your ramfs is e.g. read over the network).

But what about initramfs?

initramfs has the same inherent problems. It does a little bit better job because it copies the data to the buffer cache. After that the memory occupied by the initramfs image is released. However, still it reads more than needed.

So ram disks are useless?

No, not at all. Having populated ram disk and use them as root filesystem is often not very meaningful. However a ram disk or ramfs device which is used to store temporary data can be very useful. It is a lot faster to write a 1MB temp file to RAM than it is to write it to flash.

Category:

- Tips and Tricks

> From: eLinux.org

Reordering of driver initialization

Contents

Description

Some products need to have some type of storage available as soon as possible after kernel initialization. Think of SD card type removable media that could contain possible upgrades or other versions of a software product. In order to mount these media right away, they should be probed and recognized by the time your favorite startup script (or 'init') is started. Sometimes bus/media probing can take a long time, especially if you have slow buses, or when one bus can support several types of media. In the latter case, an SD card attached to a bus (host) that can support SD 2.0, SD 1.0 and MMC protocols, they have to be probed from super to lesser interface specification (SD cards understand basic MMC protocol as well). Worst case, the media is probed and detected after your favorite startup scrupt (or 'init') has finished its work already!

How to implement or use

By simply identifying which driver(s) is (are) associated with the medium, move them upward in the driver makefile(s) to put them more in front of the to-be-initialized kernel modules list.

Expected Improvement

Media are probed and disks are added before kernel initialization finishes, no 'hotplugging' interface necessary. Startup scripts (or 'init') can decide what to do before it is 'too late'.

Resources

None.

Projects

None.

Specifications

None.

Patches

None.

Case Studies

Categories:

- Boot Time
- Kernel

> From: eLinux.org

RTC No Sync

Contents

Introduction

One routine that potentially takes a long time during kernel startup is `get_cmos_time()`. This routine is used to read the value of the external real-time clock (RTC) when the kernel boots. Currently, this routine delays until the edge of the next second rollover, in order to ensure that the time value in the kernel is accurate with respect to the RTC.

However, this operation can take up to one full second to complete, and thus introduces up to 1 second of variability in the total bootup time.

The synchronization in this routine is easy to remove. It can be eliminated by removing the first two loops in the function `get_cmos_time()`, which is located in `include/asm-i386/mach-default/mach_time.h` for the i386 architecture. Similar routines are present in the kernel source tree for other architectures.

One tradeoff in making this modification is that the time stored by the Linux kernel is no longer completely synchronized (to the boundary of a second) with the time in the machine's realtime clock hardware. Some systems save the system time back out to the hardware clock on system shutdown. After numerous bootups and shutdowns, this lack of synchronization will cause the realtime clock value to drift from the correct time value.

Since the amount of un-synchronization is up to a second per boot cycle, this drift can be significant. However, for some embedded applications, this drift is unimportant. Also, in some situations the system time may be synchronized with an external source anyway, so the drift, if any, is corrected under normal circumstances soon after booting.

LKML discussion

The RTC synchronization was discussed on LKML in May (with even Linus himself commenting) That thread is available here.

Rationale

The RTC edge synchronization can take up to 1 second, and takes .5 seconds on average.

Downloads

Patch

- [Patch for CELF version XXXXXX is *here*]
- [Patch for 2.4.xx is *here*]
- Patch for 2.6.7 (only for i386 architecture): rtc-nosynch-3.patch
- Patch for 2.6.10 for PPC architecture: attachment:rtcnosync-ppc-2.6.10.patch

You might also want to check the Patch Archive

Utility programs

None

How To Use

Apply the patch to your Linux kernel. Then reconfigure your kernel so that "CONFIG_RTC_NO_SYNC_ON_READ" is enabled.

The option will be found under "General Setup", "Fast boot options" labeled as "No SYNC on read of Real Time Clock".

Sample Results

case 1

Tim Bird measured the effect of eliminating this synchronization on an HP xw4100 workstation. The machine had a Pentium 4 processor running at 3 GHz. With the synchronization, the time to perform get_cmos_time() varied from 150 milliseconds to 900 milliseconds. Without the synchronization, the time to perform get_cmos_time() was under 1 millisecond.

case 2

Richard Griffiths measured the effect of eliminating the synchronization on a 1GHZ Pentium III based desktop. With synchronization executing get_cmos_time() took from 200 milliseconds to a full second. Without the synchronization, the time to perform get_cmos_time() was under 1 millisecond.

case 3

Tim Bird measured the RTC read synchronization cost for a PowerPC board. The platform was a PPC440GP (ebony) board, using a Dallas Semiconductor DS1743 RTC.

The RTC read routine is in arch/ppc/syslib/todc_time.c:todc_get_rtc_time(). The loop to synchronize with the seconds edge is in arch/ppc/kernel/time.h:time_init().

I measured the time for synchronizing the clock edge several times:

- 105 ms, 139 ms, 313 ms, 572 ms, 213 ms, 626 ms, 753 ms, 426 ms, 535 ms, 163 ms

Note that for PPC, the synchronization code is *outside* the RTC read routine, instead of inside the routine (as for i386).

Status

Fixed in mainline.

Matt Mackall submitted a large patch set to fix this problem, in March

1. See http://lkml.org/lkml/2006/3/17/340 . These patches were mainlined in the Linux kernel in version 2.6.16. See http://git.kernel.org/?p=linux/kernel/git/torvalds/linux-2.6.git;a=commit;h=63732c2f37093d63102d53e70866cf87bf0c0479

Previous mainline attempts

A patch was eventually submitted to LKML for consideration (in Nov, 2004). The thread is here

There was no response.

Future Work

Here is a list of things that could be worked on for this feature:

- In an earlier thread on LKML, it was suggested that some background method of synchronizing the system clock and hardware clock could be used to avoid the clock drift problem.
- one idea is: "you could check the cmos time in the timer interrupt during boot, and correct it there rather than busy-waiting"
- Another person notes:

> "There is hwclock that will read or write the CMOS clock, and it synchronizes. So if one wants to synchronize with the CMOS clock (rather than, say, with an external clock), and wants the better-than-1-sec accuracy, then that can be done in a boot script."

- We need to check if hwclock uses get_cmos_time() to read the clock. If so, this method of synchronizing the clock edge after booting won't work (once the get_cmos_time synch is disabled.)

Categories:

- Boot Time
- Kernel

> From: eLinux.org

Short IDE Delays

Contents

Introduction

This page describes the feature "short IDE delays". This is just a description of a technique for shortening the probing time used during certain IDE operations (I think primarily on startup). It was noted on a test machine that IDE initialization takes a significant percentage of the total bootup time. Almost all of this time is spent busywaiting in the routine `ide_delay_50ms()`.

It is trivial to modify the value of the delay used in this routine.

Reducing the duration of the delay in the `ide_delay_50ms()` routine provides a substantial reduction in the overall bootup time for the kernel on a typical desktop system. It also has potential for use in embedded systems where PCI-based IDE drives are used.

In the patch shown here, the delay was modified from 50 milliseconds to 5 milliseconds. However, for particular hardware, it may be desirable to tune the delay to the lowest possible value.

This change may only be appropriate for embedded hardware. In a desktop environment, a variety of legacy hardware may be encountered which may need these relatively long delays.

Rationale

In testing on one desktop system, IDE delays accounted for about 70% of the total kernel bootup time. These delays may not be needed for proper operation of the hardware for a particular consumer electronics product.

Downloads

Patch

- Patch for 2.6.6: short-ide-delays.patch
- Patch for 2.6.8-rc2: ide-delay-2.6.8-rc2.patch

Following is what the patch looked like for 2.6.6. Recently, the ide_delay_50ms() routine was replaced with multiple independent calls to msleep(50), which makes more recent patches more difficult.

```
diff -ruN linux-2.6.6.orig/drivers/ide/ide.c linux-2.6.6-kfi/drivers/ide/ide.c
--- linux-2.6.6.orig/drivers/ide/ide.c  2004-05-09 19:32:26.000000000 -0700
+++ linux-2.6.6-kfi/drivers/ide/ide.c   2004-06-09 21:14:01.000000000 -0700
@@ -1401,10 +1401,10 @@
 void ide_delay_50ms (void)
 {
 #ifndef CONFIG_BLK_DEV_IDECS
-   mdelay(50);
+   mdelay(5);
 #else
    __set_current_state(TASK_UNINTERRUPTIBLE);
-   schedule_timeout(1+HZ/20);
+   schedule_timeout(1+HZ/200);
 #endif /* CONFIG_BLK_DEV_IDECS */
 }
```

Utility programs

None

How To Use

Apply the patch, compile the kernel, and measure the kernel boot time.

Sample Results

As an experiment, this code (located in the file `drivers/ide/ide.c`) was modified to only delay 5 milliseconds instead of 50 milliseconds.

The machine used was an HP XW4100 Linux workstation system, with the following characteristics:

- Pentium 4 HT processor, running at 3GHz
- 512 MB RAM
- Western Digital 40G hard drive on hda
- Generic CDROM drive on hdc

When a kernel with this change was run on the test machine, the total time for the `ide_init()` routine dropped from 3327 milliseconds to 339 milliseconds. The total time spent in all invocations of `ide_delay_50ms()` was reduced from 5471 milliseconds to 552 milliseconds. The overall bootup time was reduced accordingly, by about 5 seconds.

The ide devices were successfully detected, and the devices operated without problem on the test machine. However, this configuration was not tested exhaustively.

Status

Todd Poynor posted a patch for this to LKML on July 30, 2004.See the discussion thread at: http://lkml.org/lkml/2004/7/30/141

This patch was rejected by Alan Cox, Jeff Garzik and Mark Lord. See the thread mentioned above for details. The reasons amounted to:

- the IDE driver is delicate and temperamental, changing it would break things
  - This ignored the fact that the submitted patch only changed things when a config option was enabled
  - the counterargument to this is that in this situation, the patch would not get much testing by mainline users
- if users tried this out, they might have problems and they would then bug the IDE maintainers
  - Alan suggested having the option "taint" the kernel, to avoid bug reports
- newer hardware (SATA) won't have these problems and the probes should work faster (in other words, if you ignore

this problem it will go away)
- the timeouts shouldn't be correlated to each other anyway. That is, these timeouts are used by different parts of the IDE code for different operations. The fact that they currently have the same timeout value is just happenstance, and shouldn't be locked-in with a single value definition.

Alan Cox summarized things by saying: `If you want to speed this up then the two bits that the initial proposal and Jeff have sensibly come up with are`

1. Are we doing too many probes
2. Should we switch to proper reset polling

For certain cases (PPC spin up) we actually have switched to doing drive spin up this way, I certainly have no objection to doing the rest of the boot optimisation by following the standards carefully.

Future Work

Here is a list of things that could be worked on for this feature:

- make a config option for the value of the delay
- try to verify that the change is safe to use
    - check to see if these probes are used during runtime (not just bootup time)
    - test as much hardware as possible
    - ask about the safety of this on LKML

Categories:

- Boot Time
- Kernel

> From: eLinux.org

Suspend To Disk For ARM

Table Of Contents:

Contents

Description

This page shows the result of applying suspend to disk feature for ARM architecture. Related pages are:

- Sw Suspend - Information about Sw Suspend
- Sw Suspend Porting Notes - Sw Suspend quick-hack porting guide

Target Environment

Hardware Information

OSK reference board is used.

Since OSK5912 board does not have disk to save suspend image to, flash is used to store suspend image.

Software Information

linux kernel 2.6.11 based kernel is used for swsusp, and 2.6.14 based kernel is used for suspend2. Necessary patches are described in next section.

swsusp for ARM

build kernel

Below is the kernel patch stack applied on top of vanilla 2.6.11 kernel.

- patch-2.6.11-omap1 from muru.com
- swsusp-arm-2.6.11.patch
- swsusp-osk-2.6.11.patch

To build kernel, do the following:

(assuming you have cross comiler for ARM)

- enable CONFIG_PM
- enable CONFIG_SOFTWARE_SUSPEND
- enable CONFIG_SWSUSP_MTDBLOCK_FLUSH

This can be done for example by:

```
 $ make omap_osk_5912_defconfig
 $ make menuconfig

   General setup --->
      [*] Power Management Support
      [*] Software Suspend (EXPERIMENTAL)
      [*] Flush MTD Block

also, NFS is used as rootfs

   File systems --->
      Network File Systems --->
         [*] Root file systems on NFS

 $ make uImage
```

also, check that CONFIG_SYSFS is enabled. (which is default)

do swsusp

Boot OSK board with above kernel. Issue following command as super user to suspend to disk:

```
  (mount sysfs)
  $ mkdir /sys
  $ mount -t sysfs none /sys

  (enable flash as swap)
  $ mkswap /dev/mtdblock3
  $ swapon /dev/mtdblock3

  (run some applications)

  (trigger suspend to disk)
  $ echo disk > /sys/power/state
```

After the suspend to disk is triggered, messages are emitted to console, and machine would halt.

On next bootup of the board, pass below additional argument to kernel:

-
    - resume=/dev/mtdblock3

Below is an example:

```
OMAP5912 OSK # setenv bootargs console=ttyS0,115200n8 ip=dhcp root=/dev/nfs resume=/dev/mtdblock3
OMAP5912 OSK # saveenv
```

and then boot as usual.

At near end of booting kernel, system reads suspended image, and resumes.

Known Problems / FAQs

```
- I would like to suspend image to different storage
  I have only tested with mtdblock3, and in swsusp-osk-2.6.11.patch, it is hard coded (sorry...)
  For other device (such as USB storage), I have not tested.

- When the suspended image gets big, system does not resume
  This turned out to be misconfiguration of EMIFS_CS3 register at boot loader, if you are using
  u-boot. There are 2 solutions to this:
```

-
    -
        - fix EMIFS_CS3 value in u-boot and rebuild it
        - set the correct value in kernel and rebuild it

```
  u-boot fix for file u-boot-1.1.[123]/board/omap5912osk/platform.S or u-boot-1.1.4/omap5912osk/lowlevel_init.S:

 VAL_TC_EMIFS_CS3_CONFIG:
-       .word 0x88011131
+       .word 0x88013141



  kernel re-set the value for file arch/arm/mach-omap/board-osk.c

+ #define EMIFS_CS3_VAL   (0x88013141)
static void __init osk_init(void)
{
+        /* Workaround for wrong CS3 (NOR flash) timing
+         * There are some U-Boot versions out there which configure
+         * wrong CS3 memory timings. This mainly leads to CRC
+         * or similiar errors if you use NOR flash (e.g. with JFFS2)
+         */
+        if (EMIFS_CCS(3) != EMIFS_CS3_VAL)
+                EMIFS_CCS(3) = EMIFS_CS3_VAL;
```

suspend2 for ARM

build kernel

Below is the kernel patch stack applied on top of vanilla 2.6.14 kernel.

- patch-2.6.14-omap1 from muru.com
- 100-suspend2-2.2-rc14-for-2.6.14.patch from suspend2.net
- suspend2-2.2-rc14-arm-fixups.patch
- suspend2-osk.patch

To build kernel, do the following: (assuming you have cross comiler for ARM)

- enable CONFIG_PM
- enable CONFIG_SUSPEND2
- enable CONFIG_SUSPEND2_CRYPTO
- enable CONFIG_CRYPTO
- enable CONFIG_CRYPTO_LZF

This can be done for example by:

```
 $ make omap_osk_5912_defconfig
 $ make menuconfig

   Power management options --->
      [*] Suspend2 --->
         [*] File Writer
         [*] Swap Writer


   Cryptographic options --->
      [*] Cryptographic API
      <*> LZF compression algorithm

also, change the OSK system timer to MPU Timer

   System Type --->
      TI OMAP Implementations --->
         System timer ---> use mpu timer

 $ make uImage
```

do suspend2

Boot OSK board with above kernel. On bootup of the board, pass below additional argument to kernel:

-
    - resume2=swap:/dev/mtdblock3

Below is an example:

```
OMAP5912 OSK # setenv bootargs console=ttyS0,115200n8 ip=dhcp root=/dev/nfs resume2=swap:/dev/mtdblock3
OMAP5912 OSK # saveenv
```

Install hibernate script into target system, which could be obtained from suspend2 web page.

Issue following command as super user to suspend to disk:

```
  (enable flash as swap)
  $ mkswap /dev/mtdblock3
  $ swapon /dev/mtdblock3

  (trigger suspend2 to disk)
  $ hibernate
```

After the suspend to disk is triggered, messages are emitted to console, and machine would halt.

Power the board again, make sure the kernel parameter contains resume2=swap:/dev/mtdblock3.

At near the end of kernel boot, suspend2 resume operation begins, and resumes from previously suspended state.

snapshot boot for ARM

/\ * * * WARNING * * * /\ This feature is very experimental, Your boot loader might get corrupted. /\ To restore boot loader, refer osk page], "Flash Recovery Utility" section.

what is it

Snapshot boot is similar to swsusp, but with much faster start up, by kernel and boot loader working together. In swsusp, time taken to resume is not very fast, since (a) it starts at `late_initcall', (b) snapshot image is copied twice (swap -> allocated mem -> orig mem), and (c) device state transfer from active -> suspend -> resume. In snapshot boot, image is

copied directly to orig mem addr, and jumps into kernel resume point, not kernel entry point, and kernel handles device resume, and thaw processes.

build boot loader

Below is the patch stack applied on top of u-boot-1.1.4 from u-boot web page. First three patches are to build osk5912 target, rest are for snapshot boot.

- omap5912osk-fix-undef.patch
- omap5912osk-fix-cfi-flash.patch
- omap5912osk-fix-setup-val.patch
- snapshot-boot-core.patch
- snapshot-boot-arm.patch
- snapshot-boot-osk.patch

To build u-boot, do the following:

```
$ make CROSS_COMPILE=arm-linux- omap5912osk_config
$ make CROSS_COMPILE=arm-linux- all
```

install boot loader

If autoboot is set, hit any key to enter u-boot command prompt.

Below is example installation

```
OMAP5912 OSK # tftp 0x10000000 u-boot.bin

Take note of bytes that were transfered; take note of below message
Bytes transferred = 96952 (17ab8 hex)

OMAP5912 OSK # protect off 00000000 0001ffff
OMAP5912 OSK # erase 00000000 0001ffff
OMAP5912 OSK # cp.b 0x10000000 0x00000000 0x00017ab8
OMAP5912 OSK # protect on 00000000 0001ffff
OMAP5912 OSK # reset
```

Again, if you are messed up, take a look at osk page], "Flash Recovery Utility" section.

build kernel

Below is the kernel patch stack applied on top of vanilla 2.6.11 kernel. First three patches are the same patches that were used in swsusp.

- patch-2.6.11-omap1 from muru.com
- swsusp-arm-2.6.11.patch
- swsusp-osk-2.6.11.patch
- swsusp-preserve-image.patch
- snapshot-core-2.6.11.patch
- snapshot-arm-2.6.11.patch

To build kernel, do the following:

(assuming you have cross comiler for ARM)

```
$ make omap_osk_5912_defconfig
$ make menuconfig

  General setup --->

     PCCARD (PCMCIA/CardBus) support --->
        <*> PCCard (PCMCIA/CardBus) support
        <*> 16-bit PCMCIA support (NEW)
        <*> OMAP CompactFlash Controller

     [*] Power Management Support
     [*]   Software Suspend (EXPERIMENTAL)
     [*]     Flush MTD Block
     [*]     Preserve swsuspend image

also, NFS is used as rootfs

  File systems --->
     Network File Systems --->
        [*] Root file systems on NFS

$ make uImage
```

do snapshot boot

On bootup of the board, pass below additional argument to kernel:

-
    - resume=/dev/mtdblock3
    - prsv-img

Below is an example:

```
OMAP5912 OSK # setenv bootargs console=ttyS0,115200n8 ip=dhcp root=/dev/nfs resume=/dev/mtdblock3 prsv-img
OMAP5912 OSK # saveenv
```

Boot OSK board with above kernel.

Issue following command as super user to suspend to disk: (These steps are identical to swsusp)

```
(mount sysfs)
$ mkdir /sys
$ mount -t sysfs none /sys

(enable flash as swap)
$ mkswap /dev/mtdblock3
$ swapon /dev/mtdblock3

(run some applications)

(trigger suspend to disk)
$ echo disk > /sys/power/state
```

After the suspend to disk is triggered, messages are emitted to console, and machine would halt. Power off the board.

Power on the board, and hit any key to enter u-boot command prompt. From u-boot, issue snapshot boot command.

```
OMAP5912 OSK # bootss 0x00240000
```

The snapshot boot will start, and system resumes from previously suspended system state. Note that same image could be reused, (by specifying prsv-img in kernel command line) and contribute to fast startup of whole system.

Startup comparison

Below is a comparison of swusps vs. snapshot boot startup time. Time is measured using printk times, so there is some overhead due to measurement. Rootfs is nfs, and application is mplayer, running mpeg file. Application and userland is not optimized, prelink, XIP or other startup improvement techniques are not applied, so refer the data with relative time, not absolute time. Normal startup time without swsusp nor snapshot boot is about 11 seconds.

Media:mplayer-swsusp.log Media:mplayer-ssboot.log

NOTE: The latter two log files were not found on the old CELF Pulic Wiki, therefore links are broken here.

http://elinux.org/Tiny6410

http://elinux.org/Micro2440

http://elinux.org/Mini210

http://elinux.org/Tiny210

Category:

- Kernel

> From: eLinux.org

Threaded Device Probing

This page describes Threaded device probing, which is a feature which allows drivers in the Linux kernel to have their probes execute in parallel threads. One of the most time-consuming parts of the boot up sequence is the probing by device drivers for their hardware devices.

This patch was created by Greg Kroah-Hartman and with it he was able to reduce the bootup time of the kernel on one machine by about 400 milliseconds.

Original post and discussion

See this article for the original description of this and a patch.

There was discussion about this here.

Kernel Option

Greg Kroah-Hartman posted a patch to LKML on Sep 25 2006 with a config option to turn this on for the PCI bus. His post is at: lkml thread

The kernel option to turn on this feature for the PCI bus in Linux version 2.6.18-rc4-mm1 (Andrew Morton's tree) is CONFIG_PCI_MULTITHREAD_PROBE.

Status

This code was apparently never integrated into mainline. It appears to be superceded by the Asynchronous function calls work.

Categories:

- Boot Time
- Kernel

> From: eLinux.org

Tims Fastboot Tools

This page has materials to support Tim Bird's presentation "Tools and Techniques for Reducing Bootup Time". This presentation was first delivered at the Embedded Linux Conference, Europe, in November of 2008.

Contents

safe-to-call-sched-clock.patch

This patch was used by Tim for several kernel versions, to address problems with the printk-times feature on ARM platforms. It is very simple, and consists of just adding a flag to avoid calling sched_clock() too early.

This patch appears (as of kernel version 2.6.27) to be obsolete. The 2.6.27 kernel now calls cpu_clock() for printk_times. This uses a similar mechanism to flag when it is safe to call.

Here's the patch:

```
On most platforms, printk_clock() calls sched_clock(), which provides good
timestamp resolution.  However, most ARM boards use a timer source for
sched_clock() which must be initialized at boot.  If sched_clock() is called
too early, the machine hangs. This code utilizes the default jiffies-based
value for printk_clock, until told that sched_clock() is safe.  This is
almost always 0 before the clock is initialized, so this patch causes
no loss of timing data, or confusing time switchover mid-boot.

OMAP support is included.

To utilize this on other ARM platforms, just add "safe_to_call_sched_clock=1"
in the timer initialization code for your platform, when it is safe to call
sched_clock().

Signed-off-by: Tim Bird <tim.bird@am.sony.com>

ChangeLog:
    2008/02/05
    Location: alp@oak--linux-3/alp-linux--dev-3--3.1
    First changelog version.
---
 arch/arm/kernel/time.c      |    8     7 + 1 - 0 !
 arch/arm/mach-omap1/time.c  |    3     3 + 0 - 0 !
 include/asm-arm/mach/time.h |    2     2 + 0 - 0 !
 3 files changed, 12 insertions(+), 1 deletion(-)

Index: alp-linux/arch/arm/kernel/time.c
===================================================================
--- alp-linux.orig/arch/arm/kernel/time.c
+++ alp-linux/arch/arm/kernel/time.c
@@ -84,10 +84,16 @@ static unsigned long dummy_gettimeoffset
  * sched_clock().  This avoids non-bootable kernels when
  * printk_clock is enabled.
  */
+int safe_to_call_sched_clock = 0;
```

```
+
 unsigned long long printk_clock(void)
 {
-   return (unsigned long long)(jiffies - INITIAL_JIFFIES) *
+   if (likely(safe_to_call_sched_clock)) {
+       return sched_clock();
+   } else {
+       return (unsigned long long)(jiffies - INITIAL_JIFFIES) *
            (1000000000 / HZ);
+   }
 }

 static unsigned long next_rtc_update;
Index: alp-linux/arch/arm/mach-omap1/time.c
===================================================================
--- alp-linux.orig/arch/arm/mach-omap1/time.c
+++ alp-linux/arch/arm/mach-omap1/time.c
@@ -255,6 +255,9 @@ static void __init omap_init_clocksource
    setup_irq(INT_TIMER2, &omap_mpu_timer2_irq);
    omap_mpu_timer_start(1, ~0, 1);

+   /* allow calls to sched_clock now */
+   safe_to_call_sched_clock = 1;
+
    if (clocksource_register(&clocksource_mpu))
        printk(err, clocksource_mpu.name);
 }
Index: alp-linux/include/asm-arm/mach/time.h
===================================================================
--- alp-linux.orig/include/asm-arm/mach/time.h
+++ alp-linux/include/asm-arm/mach/time.h
@@ -76,4 +76,6 @@ extern int (*set_rtc)(void);
 extern void save_time_delta(struct timespec *delta, struct timespec *rtc);
 extern void restore_time_delta(struct timespec *delta, struct timespec *rtc);

+extern int safe_to_call_sched_clock;
+
 #endif
```

grabserial

"grabserial" is a simple program to grab and display data from a specified serial port. It can place a timestamp on each line received, which makes it useful for reporting timing of events seen on a booting system's serial console output.

See Grabserial for detailed information, sample output and download instructions.

Tim's quick and dirty process trace

Process trace is Tim's quick and dirty method of tracing early boot processes. As I write this (Oct, 2008), an boot tracer is being written and is available in the fastboot git tree on kernel.org. This will likely be mainlined, and should supersede this work. However, this tool has worked for me.

This is really quite simple. The "system" consists of a patch which adds printks to fork, exec and exit, and a script which parses those printks and prints information about the reported processes.

Patch Download

Here is the patch: Media: tims_process_trace.patch

This patch was written against Linux kernel version 2.6.27.

Usage

To use this patch, apply it to your kernel, with something like:

```
cd linux_src ; patch -p1 </path/to/patch/tims_process_trace.patch
```

You should also increase the size of your kernel log buffer (this is the buffer where printk messages are stored in the kernel.) I recommend a value of 18, which is 256K. This is CONFIG_LOG_BUF_SHIFT, and is found in the "General Setup" menu of the kernel configuration system.

Compile, build and install your kernel. Boot the kernel.

After booting, use dmesg to collect the messages. Note that you need to specify the (increased) size of the message buffer with the '-s' option:

```
dmesg -s 256000 >/tmp/bootlog.txt
```

Now, process the bootlog with the scripts/procgraph program.

```
linux_src/scripts/proc_graph /tmp/bootlog.txt
```

This script is somewhat badly named. It does not generate a graph, but just allows you to sort the processes by various attributes (start time, duration, idle time, etc.) Actual graphing should be added "real soon now".

Deferred initcall

Using a short patch (available for kernel version 2.6.27) it is possible to avoid running certain initcalls at bootup time. This can save time during the critical, early portion of bootup. Later, deferred initcalls can be run by triggering them from user space.

See Deferred Initcalls.

Categories:

- Boot Time
- Measuring

> From: eLinux.org

Uncompressed kernel

Booting from an uncompressed kernel might improve the boot time.

It will take longer to read an uncompressed kernel image from background storage, but there is time saved since no decompression is needed.

So whether or not an uncompressed kernel is a win with respect to boot time depends on the speed with which you can read from the background memory and the speed of your processor (as a faster CPU will require less time to decompress). So with a fast processor and slow background memory, compression might be a win, whereas with a slow processor and fast background memory, compression might be a lose.

Best strategy here is to empirically determine whether or not it is better for your system to have an uncompressed kernel or a compressed one.

Note: the ideal situation of course is if you can use DMA to load the compressed kernel in chunks and decompress the previous chunk while the next chunk is loaded.

Category:

- Kernel

> From: eLinux.org

Networking

Contents

Introduction

Embedded Linux: Networking

This page describes general Networking as it applies to Embedded Linux. For a complete list of specific pages within the Networking page category, please click, Networking.

Categorized Pages

Networking

> From: eLinux.org

Multimedia

Contents

Introduction

Here are some miscellaneous resources related to audio, video and graphics systems under Linux:

Also see the section on User Interfaces.

CELF 2.0 Specification for AVG

(more like a set of recommendations rather than a specification)

- http://elinux.org/images//8/83/Pdf.gif AVG Spec V2 http://elinux.org/images/d/da/Info_circle.png

Audio Video Working Group

Please see the CELF wiki for more information: Audio Video Graphics Working Group

Some AVWG related Outdated pages

DirectFB study

What is DirectFB, How Does DirectFB Work

DirectFB

Sample Implementation of DirectFB on an embedded Linux platform

Porting DirectFB

Some DirectFB benchmark on embedded Linux platform

Benchmark DirectFB

Related Projects

Graphics/Video out

Framebuffer

- http://www.kernel.org/ (1) KD26/fb
- http://linuxconsole.sourceforge.net/fbdev/HOWTO/
- http://www.tldp.org/HOWTO/Framebuffer-HOWTO.html

Stores the frame information in the videos ....

DirectFB

- http://www.directfb.org/
- http://www.directfb.org/documentation/DirectFB_overview_V0.2.pdf
- DirectFB

V4L2

- http://www.linuxtv.org/
- http://www.linuxtv.org/downloads/video4linux/API/V4L2_API/spec-single/v4l2.html

X11

- http://www.x.org/
- X11

NanoX

- http://www.microwindows.org/

OpenGL (OpenML)

- http://www.opengl.org/
- http://www.khronos.org/opengles/

SDL

- http://www.libsdl.org/ immediate renderer library with very bare bones primitives like rectangle fill and blit. Since it exposes just framebuffer and few primitives, it's easy to port to different platforms, actually it was born as a way to port Windows games to Linux.

Cairo

- http://www.cairographics.org/ is an immediate renderer library that can do complex vector graphics, including matrix transforms. It runs on top of DirectFB, X11, memory buffers and more. It is the base of some toolkits like GTK and applications like Firefox.

Clutter

- http://clutter-project.org/ is an object-oriented 3d canvas on top of OpenGL (or OpenGL-ES) with scene management. It is based on GLib/GObject and matches nicely GNOME platform. Many powerful Linux mobile devices will ship with Clutter-based intefaces in near future, like Intel's Moblin, Ubuntu Mobile and Maemo.

Enlightenment Foundation Libraries (EFL)

The Enlightenment Foundation Libraries contains Evas, an object-oriented 2d canvs on top of OpenGL/X11, XRender/X11, X11, FB, DirectFB, DirectDraw and more. It includes scene management and integrates with Ecore, matches nicely other EFL components like Edje. It's used by some media centers and the OpenMoko phone. See Gustavo Barbieri conference at Embedded Linux Conference Europe 2008, slides and video. Gustavo's company, ProFUSION, offers services around EFL.

Qt

- Qt is a crossplatform graphics toolkit with support for framebuffer and X. Has advanced animation capabilities using Graphics View framework.

Storyboard Suite (Storyboard Designer/Engine from Crank Software)

- http://www.cranksoftware.com/storyboard The Storyboard Suite from Crank Software provides a complete environment to design, develop and deploy embedded user interfaces across multiple rendering technologies (DirectFB, FBDev, SDL, OpenGL, OpenVG, ...), multiple operating systems (Linux, QNX, VxWorks, WinCE/Win32, ...) and multiple architectures (x86, ARM, PPC, SH, ...). The Storyboard approach is unique in that it has been developed to incorporate content directly from graphic designers and deploy data bundles that are specifically optimized for each OS/CPU/Rendering technology set.

GStreamer

- http://www.gstreamer.net/ is a open-source multimedia framework allowing the creation of multimedia applications by assembling processing nodes (called elements) in a graph (called pipeline). The range of plugins available allow easy creation of playback applications, recorders, audio/video editors, streaming servers, visioconference system. The variety of plugins range from decoders, encoders, muxer, demuxers, network sources for a variety of protocols, hardware accelerated features (decoding, display, capture,..), video filters. Its low-level flexibility also makes it sometimes complex to use, but is assisted by several convenience plugins linke playbin, decodebin, camerabin making simple use-cases easy to use. It is built on top of GLib/GObject, making it easily portable to any new platform. It is being used in more and more in embedded devices due to the availability of quality LGPL plugins for various format support, support for lip-sync, support for network streaming, standard linux API, and easy wrapability of hardware devices like DSP-accelerated codecs.

Xine

- http://xinehq.de/ is a playback media engine that handles most of the complexity for you. It's based on threads, so clock and synchronization are handled automatically. Note that this library is GPL licensed so your application must be GPL compatible to use it.

MPlayer

- http://mplayerhq.hu/ it's not a library but an application, however it's controllable from other applications and it's used as media framework for some systems. It's GPL as well as xine, but since it's externally controlled you don't need to make your application GPL to use it.

Documentation

- Presentation Choosing embedded graphical libraries held by Thomas Petazzoni at the ELCE 2008

Video in

V4L[2]

- http://www.kernel.org/ (1) KD26/video4linux
- http://www.linuxtv.org
- http://www.linuxtv.org/downloads/video4linux/API/V4L2_API/spec-single/v4l2.html

OpenML

- http://www.khronos.org/openml/

LinuxTV (DVB API)

- http://www.linuxtv.org

Audio in/out

OSS

- http://www.kernel.org/ (1) KD26/sound/oss
- http://www.4front-tech.com/oss.html

ALSA

- http://www.kernel.org/ (1) KD26/sound/alsa
- http://www.alsa-project.org

OpenAL

- http://www.openal.org/

PulseAudio

- http://pulseaudio.org PulseAudio is a multi-platform sound server which brings a lot of cool features to Linux sound. Why you want to use it for for embedded systems (besides having support for cool stuff like bluetooth audio, apple airport etc) is described here: http://0pointer.de/blog/projects/pulse-glitch-free.html

Users of AVG

Video Lan

- http://www.videolan.org

Freevo

- http://freevo.sourceforge.net

LinuxTV

- http://www.linuxtv.org/

MythTV

- http://www.mythtv.org/

DVR

- http://dvr.sourceforge.net/html/main.html

OpenPVR

- http://www.funktronics.ca/openpvr/
- http://sourceforge.net/projects/openpvr/

Morphine.TV

- http://wiki.morphine.tv
- http://sourceforge.net/projects/mms4l/

Other

ARIB architecture

- http://www.arib.or.jp/english/html/overview/ov/std_b24.html

Boot Splash

- www.bootsplash.org

Digital Home Working Group

- http://www.dhwg.org/

Disko Framework

- http://www.diskohq.org
- http://www.directfb.org

Free Type

- http://freetype.sourceforge.net/freetype2/

UPnP

- see UPnP

TV Anytime

- http://www.tv-anytime.org/

TV Linux Alliance

- http://www.tvlinuxalliance.com/

Note (1) - KD26 refers to the Linux 2.6.X kernel tree, which has a "Documentation" sub-directory.

Categories:

- Multimedia
- CE Linux Working Groups

> From: eLinux.org

Benchmark DirectFB

Benchmarks

The DirectFB example suites include benchmark 'df_dok'. We have ran this benchmark on the following platform:

Platform	CPU	Clock	I/F	System RAM	Graphics Card	Kernel Version
A	Renesas SH-4	240MHz	CPU	64MB	SMI SM501 software drawing	2.4.20 CELF
A*					SMI SM501 enable 2D acceleration	2.4.20 CELF
B	Renesas SH-4	240MHz	PCI	64MB	Matrox Millenium	2.4.19
C	Intel Celeron	450MHz	PCI	128MB	Matrox Mystique	2.4.20
D	Intel Celeron	450MHz	PCI	128MB	Matrox Millenium	2.4.20
F	Intel Celeron	450MHz	AGP	128MB	Matrox G450	2.4.20
E	Intel Pentium4	2.4GHz	AGP	1GB	Matrox G450	2.4.20
G	Intel CE2110	1Ghz	CPU	100Mb	software drawing	2.6.16
G*					Intel GDL accelerated	2.6.16

Benchmarks	Platform								
	A	A*	B	C	D	E	F	G	G*
Anti-aliased Text [KChars/sec]	29.22	29.41	20.40	24.83	23.96	607.39	750.00	84	45
Anti-aliased Text (blend) [KChars/sec]	7.52	7.47	6.12	16.52	16.66	613.00	752.85	25	19
Fill Rectangles [MPixel/sec]	14.07	221.49	63.63	116.37	53.25	892.88	849.22	34	105
Fill Rectangles (blend) [MPixel/sec]	1.64	1.67	1.20	3.18	3.26	236.38	225.84	3	6
Fill Triangles [MPixel/sec]	12.25	93.87	93.87	108.79	50.51	748.55	730.24	32	1
Fill Triangles (blend) [MPixel/sec]	1.63	1.61	1.17	3.13	3.17	223.32	218.24	4	0.026
Draw Rectangles [KRects/sec]	1.81	16.01	10.67	12.95	8.57	59.53	36.27	11	3.6

Draw Rectangles (blend) [KRects/sec]	0.52	0.57	0.43	0.83	0.84	25.40	17.09	2.142	0.194
Draw Lines [KLines/sec]	7.10	66.98	61.33	62.60	48.84	281.86	162.40	41	16
Draw Lines (blend) [KLines/sec]	2.33	2.43	1.94	3.69	3.70	127.29	80.04	10	7
Blit [MPixel/sec]	8.12	100.49	38.68	53.75	32.56	435.02	398.84	25	64
Blit colorkeyed [MPixel/sec]	1.63	102.16	39.19	58.69	32.54	445.20	421.97	28	36
Blit with format conversion [MPixel/sec]	4.04	4.13	3.59	18.11	17.79	203.38	193.26	13	38
Blit from 32bit (alphachannel blend) [MPixel/sec]	1.05	1.05	0.82	2.71	2.71	171.32	158.10	3	36
Blit from 8bit palette [MPixel/sec]	3.54	2.35	3.20	17.40	17.38	17.28	95.17	-	-
Blit from 8bit palette (alphachannel blend) [MPixel/sec]	1.02	0.90	0.81	2.67	2.71	3.55	5.53	-	-
Stretch Blit [MPixel/sec]	12.17	83.50	7.06	83.50	47.61	210.32	220.77	25	21
Stretch Blit colorkeyed [MPixel/sec]	7.46	7.43	4.20	46.17	46.30	211.97	221.64	26	20

After enabling 2D acceleration in A* environment, the graphics benchmark score improves. It looks like even embedded processor like SH-4 can be used when DirectFB can use hardware acceleration. So a graphics library would be essential in a graphical system.

For G* (Intel CE2110) environment, just basic operations like opaque rectangle and blits are really optimized, some operations are much slower and we can notice rendering artifacts/bugs. So take care, possible you'll have to fallback to software rendering on such platform. Cyan-colored cells are supposed to be accelerated.

Categories:

- DirectFB
- Multimedia
- SuperH

> From: eLinux.org

DirectFB

This page was obtained from *DirectFB Analysis by IGEL Co.,Ltd.*

Contents

What is DirectFB?

DirectFB (Direct Frame Buffer) is a set of graphics APIs implemented on top of the Linux Frame Buffer (fbdev) abstraction layer.

- Input Device and Window System are also available
- Linux version of `DirectDraw`, in a manner of speaking
- www.directfb.org
- DirectFB wikipedia entry

DirectFB is primarily developed by the German Company, Convergence GmbH

- Specialty in Linux based DTV benchmark and middleware development Hosts LinuxTV, `FusionSound` and DirectFB projects
- Member of Multimedia Home Platform (http://www.mhp.org/)
- www.convergence.de

Goals of DirectFB

- Small memory Footprint
- Maximize utilities of hardware acceleration
- Support of advanced graphics operations such as multiple alpha blending modes
- No kernel modifications
- No library dependencies, except for libc
- Meet the requirements of MHP specifications

Features of DirectFB

Graphics

- Rectangle Filling/Drawing
- Triangle Filling/Drawing
- Line Drawing
- Blit
- Alpha Blending (texture alpha, alpha modulation)
- Porter/Duff
- Colorizing
- Source Color Keying
- Destination Color Keying
- Integrated Window System
- A subset of OpenGL API (Mesa)

Devices supported by DirectFB

As of DirectFB 0.9.21

Graphics Drivers

- Matrox Mystique/Millennium, G100, G200, G400/450, G550
- Via CLE266
- ATI Mach64/Rage Pro series
- ATI Rage 128
- ATI Radeon
- 3dfx Voodoo3/4/5/Banshee
- igs `CyberPro` 5xxx
- S3 Savage 3/4 series
- `NeoMagic` 220/2230/2360/2380
- nVidia `TNT/GeForce` seiries
- SiS 315
- Intel i810
- NSC Geode

Input Drivers

- Standard Keyboards
- Serial and PS/2 mice
- joysticks
- Linux Input Layer Devices
- Infrared Remote Controller (lirc)
- iPAQ Touch Screen
- ucb 1x00 Touch Screen
- Microtech Touch Screen
- Sony PI Jogdial

Supported Media Format

Still Images

- JPEG (libjpeg)
- PNG (libpng2)
- GIF
- Image format supported by Imlib2

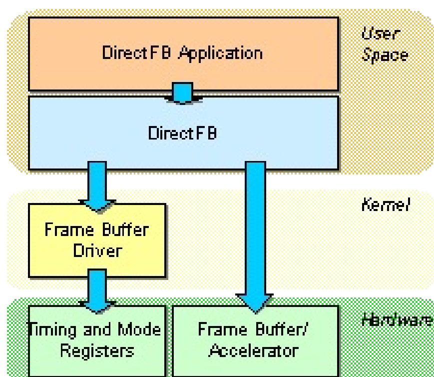
Motion Picture

- mpeg1/2 (libmpeg3)
- AVI (avifile)
- MOV (`OpenQuicktime`)
- Macromedia Flash (libflash)
- video4linux

Fonts

- DirectFB bitmap font
- `TrueType` (`FreeType2`)

DirectFB Architecture



DirectFB runs on top of the Frame Buffer Device (http://eLinux.org/dev/fb), and utilize hardware acceleration with their chipset drivers

- Also runs on VESA frame buffer

DirectFB performs only the following tasks thru /dev/fb?

- Setting up the video mode (resolution, color depth and timings)
- Memory mapping of the frame buffer and I/O ports
- Changing the viewport (e.g. for double-buffer)

DirectFB uses their own chipset driver to utilize hardware acceleration through the memory mapped I/O ports

- Turns off the frame buffer drivers acceleration features

DirectFB API

DirectFB APIs are simple APIs whose design concept seems to be referring the DirectX APIs Might be enough for STB, but further analysis is required

- Does it covers enough features?
- Performance and quality, such as flickers, when used in conjunction with `Video4Linux`
  - Based on their presentation at FOSDEM, the DirectFB has capability to synchronously overlay MPEG-2 video on DirectFB window, i.e. no flickers

Upper Layer APIs supporting DirectFB

- XDirectFB
  - X Server for DirectFB

- DirectFBGL
  - OpenGL extension for DirectFB
  - Uses Mesa/DRI
- GTK+
  - GTK+ for DirectFB
- DFB++
  - C++ Interface for DirectFB
- DFBTerm
  - Terminal runs on DirectFB
- DFBSee
  - Movie player runs on DirectFB
- DFBPoint
  - Presentation application runs on DirectFB
- MythTV
  - PVR runs on DirectFB
- Qt on DirectFB
  - Qt for DirectFB
- SDL (Simple Directmedia Layer)
  - Game Development API that also supports DirectFB
- Enlightenment Foundation Libraries (Evas and Ecore)
  - Object-oriented and stateful canvas (Evas) and main loop integration (Ecore) that have DirectFB engines/backends.
- Crank Storyboard on DirectFB
  - http://www.cranksoftware.com/products/crank_storyboard_suite.php

Fusion Sound

Audio subsystem that runs in conjunction with DirectFB Uses Fusion IPC, kernel function for synchronization, to synchronize with DirectFB

- By using Fusion IPC, DirectFB and Fusion Sound can achieve fine granularity synchronization
- Not enough documentation is available for Fusion IPC, requires source code level analysis

Consideration

Need further investigation and tests on the following points:

- How well and smoothly works with video4linux
- Internal architecture of Fusion Sound and Fusion IPC
- Coverage of features

Architecture, at a glance, seems to be well designed, however, to increase supportability of new video and audio devices, we may need to re-design the architectures, e.g. both DirectFB layer and the Linux Frame Buffer Device To accomplish inter-media synchronization, i.e. synchronization of device access and temporal synchronization, concrete synchronization subsystem needs to be designed Based on the synchronization subsystem, API similar to `DirectDraw`, `DirectSound` and `DirectShow` has to be designed (See next slide)

Development Resources

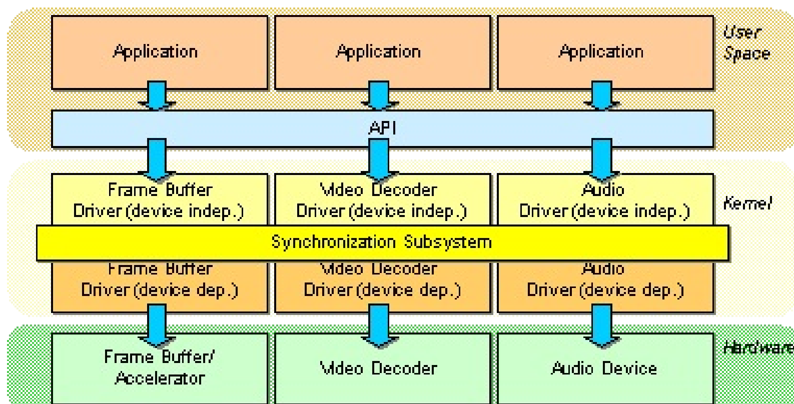
Porting Guides

See Porting DirectFB - which describes the process of porting DirectFB to an SH platform.

Performance benchmarking

See Benchmark DirectFB

Proposed System Architecture for AVWG Platform



Categories:

- DirectFB
- Multimedia

From: eLinux.org

EFL

Enlightenment Foundation Libraries, or EFL, are the set of libraries used to create the Enlightenment Window Manager DR17 (E17). This set of libraries is not restricted to X11 as Enlightenment WM itself.

- http://www.enlightenment.org/
- http://wiki.enlightenment.org/
- http://trac.enlightenment.org/e
- `svn co http://svn.enlightenment.org/svn/e/trunk/$PROJECT e`



Contents

Libraries

Most used libraries follows, you can see them all at SVN:

- Evas object-oriented and stateful 2D canvas, uses retained render mode and is very optimized, both with hardware acceleration with OpenGL and XRender and software. Ships with 32bpp and 16bpp native engines, but can downscale from 32bpp down to 1bpp with dithering on render time.
- Ecore set of libraries that covers main loop with events, including UNIX signals, timers, file descriptors, animators, pollers (shared timers). It also contains integration code with X11, DirectFB, Win32/DirectDraw, MacOS/Quartz. Extra functionality are simple IPC and HTTP communication.
- Edje is a super-theme engine, it builds on top of Evas, Ecore, Embryo, EET and others to provide powerful themes. With it your application themes are not restricted to just replace some images or changing some colors, you can

radically change the look and feel. It can be thought as a mix of HTML, CSS and JavaScript. It's based on object states and transition between theirs states, so animations can be easily specified. State descriptions can use relative and absolute positioning, so your interfaces can be scalable yet pixel perfect.

- Emotion is a media playback system integrated with Ecore and Evas. It ships with Xine, GStreamer and VLC engines.
- Epsilon is an image and video thumbnailer, builds on top of Evas, Ecore and Emotion.
- Eina basic data types, very optimized. Includes stringshare, lists, hashes and red-black trees.
- EET data storage library and command line tool, provides ar/zip-similar storage and is optimized for reading. It can handle entry compression, signing and crypto. It bundles an easy to use C-struct serialization, so one can loads complex data from files directly to dynamically allocated C structs or persist them into files. Ideal to save user preferences in an optimized way, but still editable with regular tools/editors if you use `eet -e` and `eet -e`. It's used Edje to store theme resources, as fonts, images, descriptions and even embryo scripts.
- Embryo is a tiny, simple and fast virtual machine that runs a C subset, derived from PAWN/SMALL.

Old Style Toolkits

Some more traditional toolkits were build on top of EFL but don't expose underneath technologies like Evas or Edje. They're targeted at form-like applications, just like GTK and Qt.

- ETK, similar to GTK in API. *not being actively developed anymore*
- EWL, more Model-View-Controller approach.

New Style Toolkits

Some new toolkits were born recently and makes more use of EFL concepts, not just exposing components like Evas and Edje, but truly mixing with them. So far we have two alternatives, but they will merge soon. See Rich GUI without pain ELC-E 2008 talk.

- Elementary, targeted at phones, pdas and other touch screen devices. It was originally born as a port of E17 widgets.
- Guarana, targeted at digital tv, media centers and other set-top boxes. Guarana is more than just widgets, it also covers module loader, MVC framework and more.

Demos and Videos

- E17/Illume running on Palm Treo 650
- E17/Illume running on Freescale iMX31
- Guarana demo: Enjoy Media Player
- Video of Gustavo Barbieri talk on EFL at Embedded Linux Conference Europe 2008. Slides are also available.

Category:

- Libraries

> From: eLinux.org

Outdated AVWG pages

(Redirected from Outdated pages)

Audio_Video_Graphics_Spec_5fR2

Audio_Video_Graphics_Task_Forces

> From: eLinux.org

Porting DirectFB

Contents

DirectFB porting to RTS7751R2D (SH-4/SM501) platform

Development Environment

SH-4 target platform environment

Item	Element	Resources
Hardware	Renesas `RTS7751R2D` (Renesas SH-4 processor board)	RTS7751R2D Handling Manual
	Silicon Motion Inc SM501 (Renesas platform on board graphics)	
Software	kernel = CE-LINUX-2.4.20	Tool Chains
	loader = SH-LILO	
	DirectFB = DirectFB-0.9.20	http://www.directfb.org/download/DirectFB/DirectFB-0.9.20.tar.gz
	freetype = freetype-2.1.3	http://heanet.dl.sourceforge.net/sourceforge/freetype/freetype-2.1.3.tar.gz
	libjpeg = jpeg-6b	ftp://ftp.uu.net/graphics/jpeg/jpegsrc.v6b.tar.gz
	libpng = png-1.2.2	http://keihanna.dl.sourceforge.net/sourceforge/libpng/libpng-1.2.2.tar.gz
	zlib = zlib-1.1.4	http://www.zlib.net/zlib-1.1.4.tar.gz

x86 host machine environment

Item	Element	Resources
OS	Linux (Redhat 7.3)	[1]
host compiler	gcc 2.9.6	[2]
cross compiler	gcc 3.2.3 (CELF sh-4 compiler posted by Lineo Solutions)	Tool Chains

Install DirectFB

Prepare libraries used in DirectFB

- Extract each source code
- Set environment variables for cross compile

```
CC  = /usr/local/bin/sh4-linux-gcc
CPP = /usr/local/bin/sh4-linux-cpp
CXX = /usr/local/bin/sh4-linux-g++
```

- create libjpeg

```
modify configure file for cross compile
line 548 : change "gcc" to "sh4-linux-gcc"

# ./configure --enable-shared
# make
# cp .libs/libjpeg.so* /usr/local/sh4-linux/lib
```

- create zlib

```
# ./configure --shared
modify Makefile for cross compile
line 24 : LDSHARED=/usr/local/bin/sh4-linux-gcc -shared -Wl,-soname,libz.so.1
line 28 : LIBS=libz.so.1.1.4
# make
# cp libz* /usr/local/sh4-linux/lib
```

- create libpng

```
# cp scripts/makefile.linux Makefile
modify Makefile for cross compile
line 11 : change "gcc" to "sh4-linux-gcc"
# make
# cp libpng12.so* /usr/local/sh4-linux/lib
```

- create freetype

```
# ./configure --host=sh4-linux
# make
# cp objs/.libs/libfreetype.so* /usr/local/sh4-linux/lib
# cp -r include/* /usr/local/sh4-linux/include
```

- Cross Compile DirectFB for SH-4

```
adopt patch upon DirectFb source directory
# gzip -d -c DirectFB-0.9.20-ftpath.patch.gz | patch -p0
# ./configure --host=sh4-linux \
#             --disable-sdl \
#             --with-ftpath=/usr/local/sh4-linux
#         --with-gfxdrivers=sm501
# make

If error occurs modify interfaces/IDirectFBFont/Makefile
line 86 : remove "-lfreetype"
```

- We use CELF SH-4 root_fs image posted by Lineo Solutions.
- We expect this root_fs is already bootable.
- We expect root_fs is mounted on /mnt.
- We add /usr/local/lib?/usr/lib to /etc/ld.so.conf
- Prepare directory for DirectFB installation

```
# mkdir /usr/lib
# mkdir /usr/local/share
# mkdir /usr/local/share/directfb-examples
# mkdir /usr/local/lib/
# mkdir /usr/local/lib/directfb-0.9.20
# mkdir /usr/local/lib/directfb-0.9.20/gfxdrivers
# mkdir /usr/local/lib/directfb-0.9.20/inputdrivers
# mkdir /usr/local/lib/directfb-0.9.20/interfaces
# mkdir /usr/local/lib/directfb-0.9.20/interfaces/IDirectFBFont
# mkdir /usr/local/lib/directfb-0.9.20/interfaces/IDirectFBImageProvider
# mkdir /usr/local/lib/directfb-0.9.20/interfaces/IDirectFBVideoProvider
# mkdir /usr/local/lib/directfb-0.9.20/systems
```

- copy libjpeg

```
move to libjpeg's directory
# cp .libs/libjpeg.so* /mnt/usr/lib
```

- copy zlib

```
move to zlib's directory
# cp libz.so* /mnt/usr/lib
```

- copy libpng

```
move to libpng's directory
# cp libpng12.so /mnt/usr/lib/libpng.so
# cp libpng12.so.0 /mnt/usr/lib/libpng.so.0
# cp libpng12.so.0.1.2.2 /mnt/usr/lib/libpng.so.0.1.2.2
```

- copy freetype

```
move to freetype's directory
# cp objs/.libs/libfreetype.so* /mnt/usr/lib
```

- copy DirectFB

```
move to DirectFB's directory
# cp src/.libs/libdirectfb.so /mnt/usr/lib
# cp src/.libs/libdirectfb-0.9.so* /mnt/usr/lib
# cp gfxdrivers/sm501/.libs/libdirectfb_sm501.so /mnt/usr/local/lib/directfb-0.9.20/gfxdrivers
# cp inputdrivers/joystick/.libs/libdirectfb_joystick.so /mnt/usr/local/lib/directfb-0.9.20/inputdrivers
# cp inputdrivers/keyboard/.libs/libdirectfb_keyboard.so /mnt/usr/local/lib/directfb-0.9.20/inputdrivers
# cp inputdrivers/ps2mouse/.libs/libdirectfb_ps2mouse.so /mnt/usr/local/lib/directfb-0.9.20/inputdrivers
# cp inputdrivers/serialmouse/.libs/libdirectfb_serialmouse.so /mnt/usr/local/lib/directfb-0.9.20/inputdrivers
# cp inputdrivers/sonypi/.libs/libdirectfb_sonypi.so /mnt/usr/local/lib/directfb-0.9.20/inputdrivers
# cp interfaces/IDirectFBFont/.libs/libidirectfbfont_*.so /mnt/usr/local/lib/directfb-0.9.20/interfaces/IDirectFBFont
# cp interfaces/IDirectFBImageProvider/.libs/libidirectfbimageprovider_*.so /mnt/usr/local/lib/directfb-0.9.20/interf
aces/IDirectFBImageProvider
# cp interfaces/IDirectFBVideoProvider/.libs/libidirectfbvideoprovider_*.so /mnt/usr/local/lib/directfb-0.9.20/interf
aces/IDirectFBVideoProvider
```

Prepare Demo software

Through previous steps, you have installed DirectFB to SH-4 Linux environment. To confirm everything created and installed properly use DirectFB demo program. You can download these demo program from DirectFB project website, that name is "DirectFB-examples". We used version 0.9.18. You can download from following Web site.

DirectFB-examples used in this tutorial.

As we faced some problem compiling these demo program for SH-4 Linux, we installed to x86 Linux first, then retarget to SH-4 Linux using following step.

```
Modify Makefile in src directry as
    "CC = gcc" to "CC = sh4-linux-gcc"
    "LIBADDS = -L/usr/local/lib -ldirectfb -lpthread -ldl" to
    "LIBADDS = -L/usr/local/sh4-linux/lib -ldirectfb -lpthread -ldl"
Adopt same modification to each Makefile in src/df_knuckles?src/df_neo?src/spacedream
```

Do make in src directory (If you do make at the upper directory, it seems not works correctly)

Install Demo program

Now demo program binary images was generated in src directory, you can copy manually these binary to target root_fs's your preferred location. Then copy all data file located under data directory to /usr/local/share/directfb-examples It seems you do not maintain/modift these file (just copy) except Makefile.

Benchmark program

Demo program named df_dok is DirectFB benchmark. If you run this without any options, it automatically runs set of benchmark program using graphics acceleration capability, and display result as graph. We use this benchmark score to tune DirectFB library for SM501.

Cross compiling

Like other Linux application program, DirectFB is designed to compile only on x86 Linux. We struggled this time to compile DirectFB for SH-4 Linux and finally we success this cross porting. However we think this is a kind of handwork way. When you try to run demo program on non-x86 Linux, x86 reference looks mandatory. I suggest you start DirectFB implementation from x86 Linux.

Writing DirectFB driver

Graphics driver detection procedure

Graphics chip drivers are located under /usr/local/lib/directfb-0.9.20/gfxdrivers There is no special installation procedure to add your own driver. DirectFB checks all drivers in driver_probe routine. When DirectFB detect correct driver in this process, DirectFB enables this driver. If driver does not works then move to next driver check, if all drivers does not much, DirectFB runs software mode without dedicated graphics driver. We have tentatively modified this process force to use SM501 driver.

Enabling hardware graphics acceleration capability

Drawing routine set command parameters to use 2D acceleration capability. Graphic driver returns TRUE or FALSE as return-value. If FALSE returned, DirectFB thinks hardware acceleration does not built-in, and use whole software drawing. Not all drawing routine has this acceleration detection mechanism, it is sure to define in `CheckState` routine. You can write your own driver as you like except command parameters and return value.

SM501 driver development

reference driver

We use i810 graphics driver as a reference, because it is small and looks easy to implement. Finally we need to refer other drivers because i810 has limited numbers of draw routine. If you want to support full drawing capabilities, Matrox driver may be better reference.

Drawing API prepared for SM501

API	Operations
`sm501_set_src`	Set source address for BLIT operation. You need to assign VRAM physical address to source address.
`sm501_set_dest`	Set destination physical address and color depth for BLIT and drawing operation.
`sm501_set_colorkey`	colorkey setting, it defines transparent color for colorkeyed BLIT operation. Color compare regiser is used in SM501 to enable this operation. You can define transparent color to the SM501 color compare register, then SM501 does not transfer this color when doing BLIT operation. So it looks transparent.
`sm501_set_color`	Define drawing color setting
`sm501_set_clip`	Save clipping coordinates value to the structure, and set SM501 clipping register as well
`sm501CheckState`	This bit returns hardware acceleration capability. You need to refer each graphics spec, and you also care for color format that graphics supports
`sm501SetState`	Do pre-process for acceleration
`sm501FillRectangle`	Do fill rectangle, using SM501 2D drawing engine
`sm501DrawRectangle`	Draw rectangles. SM501 does not have rectangle drawing capability, so draw four lines to draw rectangle in this driver
`sm501Blit`	Do BLIT operation, using SM501 2D drawing engine
`sm501StretchBlit`	Do Stretch BLIT To avoid SM501 2D engine stretch function limitation, we use CSC (color space conversion) function instead. Colorkeyed stretch BLIT can not be support in this driver.
`sm501DrawLine`	Draw line, using SM501 2D drawing engine. Use shortstroke command for simple vertical / holizontal line,and use line command for other kind of line.
`sm501fill_tri`	Draw triangle with filling inside. As SM501 does not have hardware triangle drawing capability,we draw triangle by software. However we can use horizontal line draw capability in SM501, so seems to achieved a little better performance than all software drawing.
`sm501FillTriangle`	Fill triangle, actual operation execute in subroutine sm501fill_tri
`driver_get_abi_version`	Returns version information
`driver_probe`	This should return check result of expected graphics chip is exist or not. At this moment, this driver retunes always TRUE (=SM501 exist)
`driver_get_info`	Returns driver's information
`sm501_release_resource`	Do close operation
`driver_init_driver`	Initialize SM501 2D engine and create function table
`driver_init_device`	Initialize device structure
`driver_close_device`	Do nothing at this moment
`driver_close_driver`	Do nothing at this moment

DirectFB porting notes

DirectFB requires display resolution capability built in frame buffer driver. So initially we modify SM501 generic frame buffer driver to support resolution and color depth setting. We did not care for pixel_clock, left/right margin and vertical/horizontal sync, because DirectFB does not works when frame buffer drivers return FALSE to these parameter. It looks no problem using without these parameters.

Initially we modify reference gfxdriver (=i810) source code so that SH-4 gcc compiler can compile properly. Actually we maintain filename, display character and source code comment to fit SM501. And commented out driver code that operate 2D graphics engine. After you compile SM501 driver, copy it to the proper location so that DirectFB can load this new driver. If DirectFB boot message does not comes driver location missing nor driver_probe process returns error.

Then merge 2D engine sample program that works. In SM501 initialization call 2D engine initialization and modify simple graphics API like line. You need to activate bit in `CheckState`, that has hardware acceleration in SM501. Remove unused program in driver_init_driver, to eliminate unexpected initialization.

Once SM501 hardware acceleration works, replace other drawing API. If SM501 command support DirectFB drawing API, just use SM501 command. If SM501 does not support it, you need to find out alternative way to implement this API. If your reference driver does not support required API, you need to refer other graphics driver and you need to add this function by yourself.

If you added new hardware drawing API, you need to enable bit in `CheckState` and need to maintain function table list. Due to the limitations of graphics controller if some hardware acceleration can not be used, you need to disable corresponded bit in `CheckState`. For example SM501 can not handle ARGB format. If BLIT source format does not much target format, SM501 can not use hardware BLIT capability. In such case you need to disable bit in `CheckState`.

To achieve better performance, we modify to use /dev/mem access instead of usual ioctl call, that was originally used in reference driver. After we maintained to read/write 2D engine register directly use /dev/mem, drawing performance increased dramatically especially `FillRect` type simple parameter command.

Finally we removed un-used function built in reference driver. For example i810 driver include UMA setting. PCI initialization does not need when SM501 connect to SH-4 using SH-BUS. So we remove these portion.

Reference

Unfortunately we can not find any useful information to DirectFB port to SH-4 Linux. So we do it using step-by-step way using printk(). For 2D acceleration portion, sample program and device document were helpful. But sometime it is difficult to understand actual drawing limitations. So x86 DirectFB is mandatory as a reference when you port DirectFB to the Linux runs on embedded CPU like SH-4.

Resources

You can download original source, patches and pre-compiled DirectFB program for `RTS7751R2D` platform

logs

- DirectFB configure logs conf_log
- DirectFB make logs make_log

source archives

NOTE: The versions listed below are very old. Newer versions are recommended.

- DirectFB DirectFB-0.9.20.tar.gz (1277kB)
- Freetype freetype-2.1.3.tar.gz (1021kB)
- libjpeg jpegsrc.v6b.tar.gz (599kB)
- libpng libpng-1.2.2.tar.gz (488kB)
- zlib zlib-1.1.4.tar.gz (177kB)

patches

- DirectFB SM501 patches DirectFB-0.9.20-sh4-sm501.patch.tar.gz (21kB)

kernel, root_fs

- CELF kernel Media:celinux.tgz | celinux.tgz (42MB) - NOTE that this file is too large to upload. See the old CELF Wiki and search for celinux.tgz or email the admin.
- root_fs for `RTS7751R2D` platform Media:rootfs.tgz | rootfs.tgz (36MB) NOTE that this file is too large to upload. See the old CELF Wiki and search for rootfs.tgz

Categories:

- DirectFB
- Multimedia
- SuperH

> From: eLinux.org

UPnP

Universal Plug and Play (**UPnP**) is a set of computer network Protocol (computing)|protocols promulgated by the UPnP Forum. The goals of UPnP are to allow Peripheral device|devices to connect seamlessly and to simplify the implementation of computer network|networks in the home (data sharing, communications, and entertainment) and corporate environments. UPnP achieves this by defining and publishing UPnP device control protocols built upon open, Internet-based communication standardization|standards.

The term UPnP is derived from plug-and-play, a technology for dynamically attaching devices directly to a computer.

Contents

Overview

The UPnP architecture allows peer-to-peer networking of Personal Computer|PCs, networked appliances, and wireless devices. It is a distributed, open architecture based on established standards such as internet protocol suite|TCP/IP, User Datagram Protocol|UDP, HTTP and XML.

The UPnP architecture supports zero-configuration Computer network|networking. A UPnP compatible device from any vendor can dynamically join a network, obtain an IP address, announce its name, convey its capabilities upon request, and learn about the presence and capabilities of other devices. Dynamic Host Configuration Protocol|DHCP and Domain Name System|DNS servers are optional and are only used if they are available on the network. Devices can leave the network automatically without leaving any unwanted state (computer science)|state information behind.

Other UPnP features include:

Media and device independence UPnP technology can run on many media that support IP including Ethernet, FireWire, IR (IrDA), power lines (Power line communication|PLC) and RF (Bluetooth, Wi-Fi). No special device driver support is necessary; common protocols are used instead.

User interface (UI) Control UPnP architecture enables vendor control over device user interface and interaction using the web browser.

Operating system and programming language independence Any operating system and any programming language can be used to build UPnP products. UPnP does not specify or constrain the design of an API for applications running on control points; OS vendors may create APIs that suit their customer's needs. UPnP enables vendor control over device UI and interaction using the browser as well as conventional application programmatic control.

Programmatic control UPnP architecture also enables conventional application programmatic control.

Extensibility Each UPnP product can have device-specific services layered on top of the basic architecture.

The foundation for UPnP networking is IP addressing. Each device must have a Dynamic Host Configuration Protocol (DHCP) client and search for a DHCP server when the device is first connected to the network. If no DHCP server is available, that is, the network is unmanaged, the device must assign itself an address. If during the DHCP transaction, the device obtains a domain name, for example, through a DNS server or via DNS forwarding, the device should use that name in subsequent network operations; otherwise, the device should use its IP address.

Protocol

Addressing

Discovery

Given an IP address, the step 1 in UPnP networking is Discovery. When a device is added to the network, the UPnP discovery protocol allows that device to advertise its services to control points on the network. Similarly, when a control point is added to the network, the UPnP discovery protocol allows that control point to search for devices of interest on the network. The fundamental exchange in both cases is a discovery message containing a few, essential specifics about the device or one of its services, for example, its type, identifier, and a pointer to more detailed information. The UPnP discovery protocol is based on the Simple Service Discovery Protocol (SSDP).

Description

The next step in UPnP networking is description. After a control point has discovered a device, the control point still knows very little about the device. For the control point to learn more about the device and its capabilities, or to interact with the device, the control point must retrieve the device's description from the URL provided by the device in the discovery message. The UPnP description for a device is expressed in XML and includes vendor-specific, manufacturer information like the model name and number, serial number, manufacturer name, URLs to vendor-specific web sites, etc. The description also includes a list of any embedded devices or services, as well as URLs for control, eventing, and presentation. For each service, the description includes a list of the commands, or actions, to which the service responds, and parameters, or arguments, for each action; the description for a service also includes a list of variables; these variables model the state of the service at run time, and are described in terms of their data type, range, and event characteristics.

Control

The next step in UPnP networking is control. After a control point has retrieved a description of the device, the control point can send actions to a device's service. To do this, a control point sends a suitable control message to the control URL for the service (provided in the device description). Control messages are also expressed in XML using the SOAP|Simple Object Access Protocol (SOAP). Like function calls, in response to the control message, the service returns any action-specific values. The effects of the action, if any, are modeled by changes in the variables that describe the run-time state of the service.

Event notification

The next step in UPnP networking is event notification, or "eventing". A UPnP description for a service includes a list of actions the service responds to and a list of variables that model the state of the service at run time. The service publishes updates when these variables change, and a control point may subscribe to receive this information. The service publishes updates by sending event messages. Event messages contain the names of one or more state variables and the current value of those variables. These messages are also expressed in XML and formatted using the General Event Notification Architecture (GENA). A special initial event message is sent when a control point first subscribes; this event message contains the names and values for all evented variables and allows the subscriber to initialize its model of the state of the service. To support scenarios with multiple control points, eventing is designed to keep all control points equally informed about the effects of any action. Therefore, all subscribers are sent all event messages, subscribers receive event messages for all "evented" variables that have changed, and event messages are sent no matter why the state variable changed (either in response to a requested action or because the state the service is modeling changed).

Presentation

The final step in UPnP networking is presentation. If a device has a Uniform Resource Locator|URL for presentation, then the control point can retrieve a page from this URL, load the page into a web browser, and depending on the capabilities of the page, allow a user to control the device and/or view device status. The degree to which each of these can be accomplished depends on the specific capabilities of the presentation page and device.

UPnP AV (Audio and Video) standards

UPnP **AV** stands for UPnP **Audio** and **Video**, and is a grouping within the UPnP standards supervised by the Digital Living Network Alliance|DLNA (Digital Living Network Alliance), (formerly: Digital Home Working Group), which is a forum of vendors and manufacturers who work in the home entertainment industry, and offer a "DLNA CERTIFIED™" branding for those products which follow their Networked Device Interoperability Guidelines. The Digital Living Network Alliance|DLNA forum members "*share a vision of a wired and wireless interoperable network of Personal Computers (PC), Consumer Electronics (CE) and mobile devices in the home enabling a seamless environment for sharing and growing new digital media and content services,*" and "*DLNA is focused on delivering an interoperability framework of design guidelines based on open industry standards to complete the cross-industry digital convergence*". On 12 July 2006 the UPnP Forum announced the release of 'Enhanced AV Specifications', this release was version 2 of the UPnP Audio and Video specifications (UPnP AV v2), with new MediaServer version 2.0 and MediaRenderer version 2.0 classes. These enhancements are created by adding capabilities to the UPnP AV MediaServers|UPnP AV MediaServer and MediaRenderer device classes that allow a higher level of interoperability between MediaServers and MediaRenderers from different manufacturers. Some of the early devices complying with these standards were marketed by Philips under the Streamium brand name. .

UPnP AV components

- **UPnP MediaServer DCP** - which is the UPnP-server (a 'master' device) that shares/streams media-data (like audio/video/picture/files) to UPnP-clients on the network.
- **UPnP MediaServer ControlPoint** - which is the UPnP-client (a 'slave' device) that can auto-detect UPnP-servers on the network to browse and stream media/data-files from them.
- **UPnP MediaRenderer DCP** - which is a 'slave' device that can render content.
- **UPnP RenderingControl DCP** - control MediaRenderer settings; volume, brightness, RGB, sharpness, and more).
- **UPnP Remote User Interface (RUI) client/server** - which sends/receives control-commands between the UPnP-client and UPnP-server over network, (like record, schedule, play, pause, stop, etc.).
    - **Web4CE (CEA 2014) for UPnP Remote UI**[1] - CEA-2014 standard designed by **Consumer Electronics Association's R7 Home Network Committee**. Web page|Web-based Protocol (computing)|Protocol and Software framework|Framework for Remote User Interface on UPnP Computer networking|Networks and the Internet (Web4CE). This standard allows a UPnP-capable home network device to provide its User interface|interface (display and control options) as a web page to display on any other device connected to the home network. That means that you can control a Computer networking|home networking device through any Web browser|web-browser-based communications method for Consumer Electronics Association|CE devices on a

UPnP home network using ethernet and a special version of HTML called CE-HTML.

- **QoS (Quality of Service)** - is an important (but not mandatory) service function for use with UPnP AV (Audio and Video). Quality of Service|QoS (Quality of Service) refers to control mechanisms that can provide different priority to different users or data flows, or guarantee a certain level of performance to a data flow in accordance with requests from the application program. Since UPnP AV is mostly to deliver streaming media that is often near real-time or real-time audio/video data which it is critical to be delivered within a specific time or the stream is interrupted. Quality of Service|QoS (Quality of Service) guarantees are especially important if the network capacity is limited, for example public networks, like the internet.
    - Quality of Service|QoS (Quality of Service) for UPnP consist of **Sink Device** (client-side/front-end) and **Source Device** (server-side/back-end) service functions. With Class (computer science)|classes such as; **Traffic Class** that indicates the kind of traffic in the traffic stream, (for example, audio or video). **Traffic Identifier (TID)** which identifies data packets as belonging to a unique traffic stream. **Traffic Specification (TSPEC)** which contains a set of parameters that define the characteristics of the traffic stream, (for example operating requirement and scheduling).**Traffic Stream (TS)** which is a unidirectional flow of data that originates at a source device and terminates at one or more sink device(s).

NAT traversal

One solution for NAT traversal|NAT (Network Address Translation) traversal, called the Internet Gateway Device Protocol|Internet Gateway Device (IGD) Protocol, is implemented via UPnP. Many router|routers and firewall|firewalls expose themselves as Internet Gateway Devices, allowing any local UPnP controller to perform a variety of actions, including retrieving the external IP address of the device, enumerate existing port mappings, and adding and removing port mappings. By adding a port mapping, a UPnP controller behind the IGD can enable traversal of the IGD from an external address to an internal client.

Problems with UPnP

Lack of Authentication

The UPnP protocol does not implement any authentication, so UPnP device implementations must implement their own authentication mechanisms, or implement the Device Security Service.[2] Unfortunately, many UPnP device implementations lack authentication mechanisms, and by default assume local systems and their users are completely trustworthy.[3][4] Most notably, Routers and firewalls running the UPnP IGD protocol are vulnerable to attack since the framers of the IGD implementation omitted to add any standard authentication method.

For example, Adobe Flash programs are capable of generating HTTPU (HTTP over UDP) requests. This allows a router implementing the UPnP IGD protocol to be controlled by a malicious web site when someone with a UPnP-enabled router simply visits that web site.[5] The following changes can be made silently by code embedded in an Adobe Flash object hosted on a malicious website:[6]

- Port forwarding|Port forward internal services (ports) to the router external facing side (i.e. expose computers behind a firewall to the Internet).
- Port forwarding|Port forward the router's web administration interface to the external facing side.
- Port forwarding to any external server located on the Internet, effectively allowing an attacker to attack an Internet host via the router, while hiding their IP address.
- Change Domain name system|DNS server settings so that when victims believe they are visiting a particular site (such as an on-line bank), they are redirected to a malicious website instead.
- Change the Domain name system|DNS server settings so that when a victim receives any software updates (from a source that isn't properly verified via some other mechanism, such as a checking a Public key certificate|digital certificate has been signed by a trusted source), they download malware|malicious code instead.
- Change administrative credentials to the router/firewall.
- Change Point-to-Point Protocol|PPP settings.
- Change IP address|IP settings for all interfaces.

- Change WiFi settings.
- Terminate connections.

This only applies to the #NAT traversal|"firewall-hole-punching"-feature of UPnP; it does not apply when the IGD does not support UPnP or UPnP has been disabled on the IGD.Template:Fact Also, not all routers can have such things as DNS server settings altered by UPnP because much of the specification (including LAN Host Configuration) is optional for UPnP enabled routers[7].

Other Issues

- UPnP uses HTTP over User Datagram Protocol|UDP (known as HTTPU and HTTPMU for unicast and multicast), even though this is not standardized and is specified only in an Internet-Draft that expired in 2001. [1]
- UPnP does not have a lightweight authentication protocol, while the available security protocols are complex. As a result, some UPnP devices ship with UPnP turned off by default as a security measure.

Future developments

The standard Devices Profile for Web Services|DPWS is a candidate successor for UPnP. It solves many of the problems of UPnP. A DPWS client is included in Microsoft Windows Vista as part of the Windows Rally technologies.

Another alternative, NAT Port Mapping Protocol|NAT-PMP, is an IETF draft introduced by Apple Inc in 2005.

Books

- Golden G. Richard: Service and Device Discovery : Protocols and Programming, McGraw-Hill Professional, ISBN 0-07-137959-2
- Michael Jeronimo, Jack Weast: UPnP Design by Example: A Software Developer's Guide to Universal Plug and Play, Intel Press, ISBN 0-9717861-1-9

External links

- UPnP™ Forum Universal Plug and Play Device Standards

Documentation

- UPnP™ Forum
- DLNA (Digital Living Network Alliance)
- The Jini, Vision
- technique comparison
- Microsoft WHDC UPnP webpage & links
- Universal Plug and Play in Windows XP
- Programmatically Controlling a UPnP-Capable Firewall is a document providing some basic information about coding UPnP software controllers (VBScript example source code included).
- Hacking with UPnP
- DLNA certified: how your computer, cellphone, games console, media streamer and other devices can play nicely together

News

- Vulnerability Note VU#347812 - UPnP enabled by default in multiple devices at United States Department of Homeland Security - Computer Emergency Readiness Team (Wednesday, 9 April 2008).

- Security firm predicts Microsoft Windows UPnP exploit by the end of the week at The Inquirer (Wednesday, 11 April 2007).
- Microsoft security updates for April 2007 to fix the above Microsoft Windows UPnP security issue.
- How to use Flash and UPnP to punch holes in most home firewalls at GNUCITIZEN (Saturday, 12 January 2008).

Software

UPnP Port Works (alias UPnPW) is a software implementation to configure UPnP devices via commandline.

GUPnP is an object-oriented open source framework for creating UPnP devices and control points, written in C using GObject and libsoup.

Portable SDK for UPnP Devices provides an API and open source code for building control points, devices, and bridges compliant with UPnP Device Architecture Specification v1.0 and support operating systems like Linux, *BSD, Solaris and others.

Barracuda UPnP Device and Control Point SDK for embedded devices.

Unplug n' Pray Utility to disable unnecessary UPnP servers running on home Windows machines.

Coherence Some free DLNA/UPnP tools (MediaServer/MediaRender) with a python framework. Running on Linux/BSD/Windows

AdoubleU IntelligentShare UPnP SDK for J2SE / J2ME / MIDP 2.0 Running on Linux/BSD/Windows/Mobile Devices

BRisa BRisa is written in Python for Internet Tablet OS or other Unix platforms. It enables to create MediaServer/MediaRenderer devices allowing users to share and search content from UPnP A/V devices. It will offer a plugin architecture enabling new services such as Flickr to be added as UPnP services.

J. River Media Center includes a UPnP server (aka UPnP Device) for its library.

1. ↑ Template:Cite web
2. ↑ Template:Cite web
3. ↑ Template:Cite web
4. ↑ Template:Cite web
5. ↑ Template:Cite web
6. ↑ Template:Cite web
7. ↑ Template:Cite web

Category:

- Standardized APIs

> From: eLinux.org

User Interfaces

The GUI framework is the software library which allows applications to put pixels onto the LCD. It handles contention between applications, deals with windows and usually provides an abstraction away from pixel-by-pixel drawing through widget libraries.

On a desktop system, this is separated into several components: An X-server which allows client applications to connect and draw things in the client's window. Then there are the widget toolkits, where you generally have a choice between Motif, GTK+, QT or Enlightenment (there are many others, but these are the more common ones). On an embedded system, this separation is not always as clear cut.

Contents

Global overviews

- LinuxDevices has a more detailed guide to embedded graphics here, although this is quite old now.

- Thomas Petazzoni gave a conference giving general information about the various graphical libraries available for embedded Linux devices, at Embedded Linux Conference Europe 2008. Slides and video are available.

Back Ends

Frame buffer (FBDev)

Most Linux systems on embedded devices only provide a frame buffer device. This allows an application to memory-map the pixels on the screen and write to them as if it was a regular array. This frame buffer interface is universal across all architectures and all devices, allowing applications utilizing the frame buffer to be ported easily. The disadvantage is that quite often the Application Processor has more advanced graphics hardware for rendering which is not utilized by the frame buffer interface. Another disadvantage is when contention for the frame buffer exists. What happens when two applications want to write to the framebuffer at once? How do they know not to overwrite each other's contents?

DirectFB

DirectFB addresses the issue of not utilising hardware acceleration by providing applications with a more advanced programming interface. Common graphics operations such as line drawing, blit'ing and basic windowing are provided. When the underlying hardware is able to accelerate these operations, this acceleration is used. When hardware acceleration is not available, the operation is performed in software, providing maximum compatibility for applications. While this is an excellent project, the number of hardware-accelerated devices supported are small and does not include any CPUs you'd find on a phone. This may, of course, change in the future. DirectFB also addresses the contention issue by providing windowing. Applications request a window from DirectFB, which ensures no applications overwrite each other's contents (Similar to X).

See DirectFB article.

Nano-X (Formally Microwindows)

An alternative to using a frame buffer device is to use an x server as you would on a desktop PC. There are many (MANY) applications out there which use X. The problem with x servers are their footprint. Both memory and storage requirements of a normal x server are too large for small embedded systems. To address this, an x-like server has been produced called Nano-X. While not conforming to the X windows API, the API is similar, allowing X windows applications to be ported. The last item of news on the nano-x site is dated 2005, so this project may be dead.

KDrive (formally Micro-x)

Unlike nano-x, KDrive is a complete X server and even supports the RENDER X windows extensions. Although KDrive is a full X Server, it goes to great lengths to not become bloated and is designed with embedded systems in mind. As KDrive supports the RENDER extensions, accelerating drawing operations in hardware is possible, but requires modification for the target Application Processor's graphics core. As with DirectFB, the list of supported hardware accelerated devices does not include graphics cores found in phones. There is an x server for the HP IPaq, however this simply uses the standard frame buffer and adds support for the VGA-Out device. Maemo uses Xomap, an accelerated KDrive for OMAP platforms, namely TI OMAP 1710 and 2420.

See X11 article for more information.

OpenKODE

OpenKODE is a new specification for mobile devices and includes OpenGL ES, OpenVG, OpenMAX and similar into a single, integrated API. The emphases here is hardware acceleration. OpenKODE is cross platform and is gathering a lot of industry support. One reason why this might be of interest is rather nicely demonstrated by nVidia in this user interface demo which uses an OpenKODE 1.0 implementation: Next-Gen Phone Interface. There are currently no open source implementations of the OpenKODE specification, however there are OpenGL ES & OpenVG (binary) Linux drivers available

for PowerVR MBX & SGX cores, found in higher-end [Application Processors]. These drivers are not available from Imagination Technologies (who designed the PowerVR series) directly but are instead supplied with the BSP for your chosen CPU.

Widget Toolkits

A widget toolkit provides useful widgets such as buttons, menus, text boxes etc. The application programmer doesn't care what a button looks like, just what happens when it's pressed. On embedded systems, you generally have two choices: GTK+ or QT. Recently however, people working on the E17 (Enlightenment) window manager project have started looking at running enlightenment on embedded devices. Also emerging are more advanced, Flash based user interfaces. Sadly though, there are currently no open source equivalents at present.

GTK+

GTK+ has excellent support on embedded systems. Ports of GTK+ have been made for rendering directly to the frame buffer, to a DirectFB device and of course KDrive. For more information about GTK+, visit [1].

QT/Qtopia

QT is a commercial product written by a company called Trolltech that has been bought by Nokia in January 2008. Written in C++ and providing a C++ API, Qt is not only a graphical library, but a full development framework : database access, XML, threads, data structures, networking, etc.

Until version Qt 4.5, Qt was distributed using a dual-licensing scheme :

- A version of the library distributed under the GPL. As applications using a library are considered derived works of the library, applications using the GPL version of Qt must be released under the GPL. While this is fine for the free software and open source community and some companies, it may not suit others companies needs.
- A version of the library under a commercial license, allowing the creation of proprietary products.

Since Qt 4.5, Qt is also distributed under the LGPL license, which also proprietary applications without buying a commercial license.

QT as it stands is unsuitable for embedded devices due to it's large size and large dependency tree. However, a few years ago, Trolltech launched an embedded version of QT, cleverly named Qt/Embedded. While this is still used for the OPIE application framework, Troltech have updated Qt/Embedded and produced their own framework called Qtopia. Out of the box, both Qt/Embedded and Qtopia render directly to the frame buffer. In Qtopia, it is possible to accelerate this rendering by re-implementing several of the classes the library uses to perform all drawing operations. The Both QT/Embedded & the Qtopia framework also handles windowing internally when multiple applications are running. More recently, Trolltech has incorporated OpenGL ES into Qtopia, which is used for hardware accelerated drawing operations and for compositing multiple windows. The Trolltech documentation also hints at visual effects similar to those on Beryl/Compiz, although there are no screen-shots avaliable.

Enlightenment

Enlightenment started life as a window manager for X11. Over the years, it has transformed into a very rich, full desktop environment similar to KDE and Gnome. Enlightenment has always had a very slick user interface with lots of eye-candy. The latest development version of Enlightenment, E17 is highly modularised. E17 has its own widget toolkit called the Enlightenment Widget Library (Ewl). The Ewl draws its widgets using the Enlightenment Foundation Library (ELF) which renders onto a canvas component called EVAS. EVAS supports compositing with true alpha-blending allowing some very attractive user interfaces. EVAS also supports multiple back-ends including X11, OpenGL, and the Linux framebuffer.

See Enlightenment Foundation Libraries article.

Quantum Step

QuantumStep is essentially OpenStep for mobile devices. It's layered on top of X11 so is (in theory) fairly portable. Developers may be familiar with another product derived from NeXTStep... MacOS X. QS uses the MacOS X development chain, including Xcode and Objective-C. So... if you're a Mac-o-phile, then there is an open alternative to the iPhone.

WxEmbedded

WxEmbedded is the embedded variant of WxWidgets. It is a widget library on top of X11, GTK+, DirectFB, Nano-X, MicroWindows, MGL and WinCE. Applications are portable between those platforms and get the native look and feel on them.

WxWidgets has a nice cross platform GUI programming assistants called [[2]]. It is unclear how usable this tool is in combination with WxEmbedded, but if you are interested in WxEmbedded, DialogBlocks is definitely worthwhile investigating.

Application Frameworks

The widget toolkits provide facilities for applications to draw buttons on the screen, but what if they want to find a telephone number? On a typical phone, lots of different applications are going to need a contact list. Having a different contact list for each application is going to get silly. Instead, there are several application framesworks available. These can be split into 3 groups, those based on QT, those based on GTK+ and those based on Enlightenment. This is similar to the KDE vs Gnome vs Enlightenment dilemma, just on embedded devices. In general, GTK+ frameworks use Kdrive as an X server and the matchbox window manager. Frameworks based on QT (there are only 2 of them) use the windowing system built into QT/Embedded (now Qtopia Core). As such, an application using a GTK+ framework can't run simultaneously with an application using a QT based framework.

The last 12-months has seen interest in Embedded Linux explode. As a result, the market has become fragmented, with lots of different competing application frameworks. All of these newer frameworks (OpenMoko, GPE, GPE Phone Edition, Hiker, Sato, Sugar & Hildon) are based on the GTK+/KDrive/Matchbox combination. All of these projects have forked GTK+ and develop on separate branches, which is a lot of duplicated and wasted effort. To address the problem of fragmentation, the Gnome Mobile Initiative was founded to bring these different frameworks together and introduce some consistency & compatibility. The Gnome Mobile website is at http://www.gnome.org/mobile.

GTK+ Based Frameworks

OpenMoko

OpenMoko is an open source project sponsored by FIC. It aims to create a complete open source software stack for mobile phones. As the work is being sponsored by FIC, the first phone OpenMoko is designed to run on is the FIC Neo1973. OpenMoko uses KDrive to provide the back end for graphics. All the widgets used on the phone are designed specifically for OpenMoko and written using the GTK+ toolkit.

GPE Palmtop Environment

From GPE's homepage: "*The GPE Palmtop Environment aims to provide a Free Software GUI environment for palmtop/handheld computers running the GNU/Linux™ operating system. GPE uses the X Window System, and the GTK+-2.6 widget toolkit.*

GPE is not a single piece of software, but an entire environment of components which make it possible to use your GNU/Linux handheld for standard tasks such as Personal Information Management (PIM). GPE makes it easy for developers to create powerful programs, by giving them the infrastructure they need.

Besides providing core software such as shared libraries, and perhaps more importantly, the GPE environment fixes standards for program interaction, such as SQL, XML, and other data schemas."

A recent development has been the launch of GPE Phone edition. This aims to implement a complete LIPS compliant software stack. For more information on GPE in general, visit [3]

Hiker (Access Linux Platform)

The Access Linux Platform contains an application framework called Hiker. Hiker is built on top of GTK+ and provides facilities to allow applications to exchange data etc. More information at the Access Linux Platform's website: www.access-company.com/products/linux/alp.html

Sato

Sato is another framework based on GTK+ and is produced by OpenedHand for their "Poky" Linux distribution. Sato includes lots of applications such as games, a web browser, contact list etc. See pokylinux.org for details.

Hildon (Maemo)

Hildon is user interface part of the Maemo distribution used on Nokia web tablets. It has started to be ported to other platforms such as Ultra-Mobile computers (E.g. Samsung Q1) as part of the Ubuntu Mobile & Embedded project. It, again, is based on GTK+ and contains widgets for menus, status bars etc. The best place to look into Hildon is through the http://maemo.org website.

Sugar (OLPC)

Sugar is the user interface used in the One Laptop Per Child project. It takes a different approach to the "desktop" metaphor and uses its own unique user interface. The Sugar wiki can be found at http://wiki.laptop.org/go/Sugar.

QT/Qtopia Based Frameworks

Qtopia

from Qtopia's homepage: "*Qtopia is unrivaled as the application platform and user interface for Linux, allowing efficient creation of mobile and embedded devices.*"

Qtopia is split up into different versions, including a Qtopia Phone Edition. Qtopia core is name given to the actual application framework. Apart from several demos and examples, Qtopia cores does not come with any applications. Applications are included in the "Open Source Edition" and "Phone Edition". For more information, visit [4].

Open Palmtop Integrated Environment (OPIE)

Originally started as an alternative GUI for the Sharp Zaurus, OPIE has now spread across several devices and has a large collection of applications. Opie uses an older version of QT/Embedded which lacks many of the feature of the more modern Qtopia. However, due to it's stability & large application base it remains a very interesting prospect. More information at [5]

Enlightenment Based Frameworks

Currently, there are many distinct projects using Enlightenment on embedded devices. the developers have been working with embedded devices in mind and there are numerous blog posts about things which have been prototyped.

Fancypants

Fancypants is an "advanced graphics and high-performance multi-media platform for developing embedded applications in consumer, commercial and industrial devices." It is developed commercially by Fluffy Spider Technologies. It uses EVAS and Ecore from the Enlightenment project and includes a very attractive user interface. See http://www.fancypants-graphics.com/ for details, including videos showing off some of the features.

OpenInkpot

OpenInkpot is a "project for creating a free and open-source Linux distribution for eink-based devices". It is a non-commercial project developing firmware for different e-ink readers, using Evas and Ewl for the software. Currently in early stages. See the project page for further details.

Illume

Illume is an Enlightenment DR17 module to make the window manager friendly for phones and PDAs. It's in use in OpenMoko and FreeSmartPhone - FSO. This module is integrated in SVN as part of E17 modules.

Guarana

Guarana is a framework on top of Enlightenment Foundation Libraries to help with Rich GUI, it ships with widgets targeted at set-top boxes and media centers (will merge with Elementary, see EFL), as well as Model-View-Controller framework and a plugin system. It's being used in Digital TV set-top boxes and Enjoy media player.

Others

Disko

Disko is a LGPL-licensed application framework, that can be used to develop GUI applications for embedded devices. It is closely connected to the DirectFB hardware abstraction library and specialised in touchscreen and remote control navigation. It provides a plugin based management component, which enables easy writing and extending embedded applications. Media playback (DVD, DVB, IPTV and audio streams) is provided by the well known xine media library. Disko has been ported to the TI DaVinci chipset together with Direct FB and runs on x86 as well. Disko homepage

Crank Software's Storyboard UI Suite

The Crank™ Storyboard™ UI Suite is a graphical development tool and runtime engine designed for the requirements of the embedded consumer market. Crank Storyboard Designer enables user interface (UI) designers to easily prototype the look and feel of a product, and then move the prototype directly to the embedded target for deployment. The Crank Storyboard Embedded Engine is the runtime component that drives the content developed in Crank Storyboard Designer on embedded devices. It is architected exclusively to address the unique challenges of bringing a rich user interface (UI) user experience (UX) to resource-constrained embedded devices.

Category:

- Categories

From: eLinux.org

X11

X11 also known as X.org, XFree or XServer is the most used graphics on Linux, at least on desktop.

Contents

Architecture

X11 works in a client-server architecture with communication going via fast UNIX Sockets if local or TCP/IP if remote, it's totally transparent to users.

Protocol is simple and optimized, often used via libraries like Xlib or Xcb (asynchronous). Most common commands like mouse movement and expose events are small so they don't impact too much. Since communication happens using file descriptors (either local unix sockets or tcp) one can easily integrate it in event loops (or main loops) with easy polling with poll(2) or select(2).

Extensions and Hardware Acceleration

It is extensible and can make use of hardware acceleration. With extensions like XRender one can optimize 2d rendering paths. With Xvideo it's possible to use extra planes and also pass through raw YUV data. Others like Composite can allow a composite manager to do actual drawings, possible adding nice effects like semi-transparent windows or shadows. One can use OpenGL (and OpenGL-ES) with X11.

One extension that worth special note is XShm, or the shared memory extension, that is can be used to avoid sending images or other heavy data over the wire. When using XShm images, one just need to send the image identifier and other image parameters, not the image pixels. This, however, have the impact that image creation is more expensive since Linux Kernel needs to zero memory to avoid data disclosure, but it's negligible.

Strong Points

Recent efforts are being made to optimize X11 more and its Linux integration even better, like Kernel Mode Setting, which will avoid flickers during system boot and also reduce X11 server complexity.

Although one can write directly to wire or use low level libraries like Xlib or Xcb, usually one write X11 applications using various helpers and toolkits like GTK, Qt, Evas/EFL, FLTK and more. These libraries are the base of most graphical user interfaces available for Linux, including big projects as Firefox and Thunderbird, Pidgin Instant Messenger, GNOME Desktop, KDE Desktop, Enlightenment Desktop and more. These applications and libraries are primarily developed with focus on X11, so they're always up to date and require no porting at all and receive much more testing. This ready/availability of most famous applications is the strongest point of using X11, even on embedded systems as showed by Maemo, OpenMoko, OLPC and more.

Weak Points

X11 was for a long time neglected from embedded systems tagged as slow and big. While not the very truth, it do have some true facts.

- X11 is usually bigger than alternatives like raw framebuffer and DirectFB, but it's not too bad is you consider minimum

system, see Thomas Petazzoni's talk at ELC-E 2008: Choosing embedded graphical libraries.
- Due chicken-egg problem and apparently bit more complicated api for X drivers, embedded hardware manufactures provide more drivers for DirectFB than X11.

There are benchmarks that show that software-on-software comparison X11 does not add much overhead compared to DirectFB and raw framebuffer: http://profusion.mobi/node/11

Category:

- Multimedia

> From: eLinux.org

System Size

Contents

Introduction

Here are some links to information and projects related to Linux system size.

Technologies for decreasing system size

Kernel size reduction

Another wiki at https://tiny.wiki.kernel.org/ has information about renewed efforts (as of August 2014) to reduce the kernel size.

Configuration Options

- Kernel Size Tuning Guide - document about measuring kernel size and configuring the kernel for smallest size

The Linux-tiny patchset

- The Linux Tiny patch set is a collection of patches which can be used to make the Linux kernel consume less space. The long-term goal of the Linux-tiny project is to mainline these patches. Several patches have been mainlined over the last few years, and work continues in this area.

"dietnet"

Andi Kleen submitted a set of patches (May 2014) to reduce the size of the Linux kernel networking stack. See the submission thread here: https://lkml.org/lkml/2014/5/5/686

Andi states that the patches support 3 use cases:

- full-featured network stack (default Linux network stack)
- client-only stack - with reduced features but still compatible with normal user-space programs, and suitable for some uses
- minimal subset for deeply embedded, which would require specialized user-space software

In order to get full size reductions, it is best to use these patches with LTO. Doing so results in network stack that requires about 170K to run (versus 400K for the default stack).

Compiler options for reducing kernel size

An LWN article talks about three gcc options to shrink the kernel.

Shrinking the Kernel with GCC

The first option is -Os which is already in the tiny kernel patch.

Since version 3.4, gcc offered a -funit-at-a-time option. This apparently made gcc do a much better job of inlining and dead code removal, reducing the size of both text and data. It depended on another inlining patch. According to gcc's manual, this option no longer does anything.

The -fwhole-program --combine option set is equivalent to grouping all source files and making all variables static. These options are still supported by gcc, but not longer offered in BusyBox configuration options. What happened?

Another option, -mregparm=3, seems to be x86 specific, it instructs the compiler to use registers for the first three function arguments. by John Rigby

See [1] for all available optimization switches. See Compiler_Optimization for more details on effects of optimization options.

Section garbage collection patchset

These patches can shrink kernel size by ~10% by improving dead code/data elimination at link time. They are being pushed to mainline. Due to a linker bug, their acceptance depends on a newer, fixed linker (will be in binutils-2.21). Good news are that the bug affects only certain architectures (parisc), so the patches are usable even with "old" linker.

Runtime size of kernel

Often, the focus of memory size reduction for the kernel is on the size of the statically compiled image for the kernel. However, the kernel also allocates memory dynamically when it runs. On loading, the kernel creates several tables for things like network and file system structures.

Here is a table showing different kernel hash tables, and their approximate size for a 2.6 kernel. (Table taken from page 25 of http://logfs.org/~joern/data_structures.pdf)

Hash Table	memory < 512MiB RAM	memory >=512MiB RAM
	32b/64b	**32b/64b**
TCP established	96k/192k	384k/768k
TCP bind	64k/128k	256k/512k
IP route cache	128k/256k	512k/1M
Inode-cache	64k/128k	64k/128k
Dentry cache	32k/64k	32k/64k
Total	384k/768k	1248k/2496k

kernel stack size

There used to be a configuration option for reducing the size of the kernel stack for each process to 4K. By default (as of 2011), the default kernel stack size is 8K. If you have a lot of processes, then using 4K stacks can reduce the kernel stack usage.

Some notes about this are at: Kernel Small Stacks

Auto-reduction

In 2012, Tim Bird studied a few different techniques related to automatic size reduction and whole-system optimization. Specifically, he studied the following items:

- link-time optimization of the kernel
- syscall elimination
- global constraints
- kernel stack reduction

Tim also found some very interesting academic research on link-time re-writing and cold-code compression. Tim's work was presented at LinuxCon Japan in May, 2013.

A draft outline and completed slides for the talk are at System Size Auto-Reduction

Compressed printk messages

The open project proposal Compressed printk messages evaluated this technique in 2014. The results can be found on the Compressed printk messages - Results page.

Reduction Ideas and recent work

A group of developers is (as of 2014) doing continued size reduction work on the Linux kernel. A page has been set up to categorize recent work and ideas for future kernel size reductions. The page is: Kernel Size Reduction Work

File system compression

For read-only data, it is useful to utilize a compressed file system. The following are used heavily in embedded systems:

- Cramfs and SquashFS, for block storage.
- JFFS2 and its successor UBIFS, for flash (MTD) storage.

Note that Cramfs and Squashfs, due to their "write-only-once" nature, can also be used on MTD storage.

See the File Systems page for more information.

Shrinking your application

Compiler options for program size

You can use "gcc -Os" to optimize for size.

Stripping your program

You can use the 'strip' command to eliminate unneeded symbols from your application. The 'strip' command should be included with your toolchain, and may be architecture-specific. (I.e. you may need to run it with a toolchain prefix, like "arm-linux-strip")

Note that this makes debugging your application more difficult, because the debug symbols are no longer available.

By default, strip just removes debug symbols. You can remove everything but the essential symbols used for dynamic linking. To get the highest savings, use "strip --strip-unneeded \"

This can save a lot of space, especially if debug symbols were included in the build.

```
$ gcc -g hello.c -o hello
$ ls -l hello
-rwxrwxr-x 1 tbird tbird 6143 2009-02-10 09:43 hello
$ strip hello
$ ls -l hello
-rwxrwxr-x 1 tbird tbird 3228 2009-02-10 09:43 hello
$ strip --strip-unneeded hello
$ ls -l hello
-rwxrwxr-x 1 tbird tbird 3228 2009-02-10 09:43 hello
```

Now, compiles without debug symbols to start with:

```
$ gcc hello.c -o hello
$ ls -l hello
-rwxrwxr-x 1 tbird tbird 4903 2009-02-10 09:45 hello
$ strip hello
$ ls -l hello
-rwxrwxr-x 1 tbird tbird 3228 2009-02-10 09:45 hello
```

You can strip both executables as well as shared libraries.

There is a "super-strip" utility, which removes additional material from an ELF executable program (which 'strip' usually misses). It is available at: http://muppetlabs.com/~breadbox/software/elfkickers.html *This program appears to be obsolete now. I couldn't get it to compile on Fedora 8.*

Some information about stripping individual sections by hand, using the -R command is available at: http://reverse.lostrealm.com/protect/strip.html

Hand-optimizing programs, for size

If you are very intent on creating small binaries, you can use some techniques to manually create the smallest Linux executables possible.

See A Whirlwind Tutorial on Creating Really Teensy ELF Executables for Linux

Library savings

Use of a smaller libc

Glibc is the default C library used for Linux systems. Glibc is about 2 meg. in size. Other C libraries are also available for Linux, and they offer varying degrees of compatibility and size savings. In general, uClibc is considered a very good alternative to glibc, for systems where size is an issue.

- uClibc - small footprint but complete C library
- dietlibc - another library to produce very small statically compiled executables.
- klibc - very small library for use in init ram filesystems
- eglibc - a version of glibc designed for embedded systems. Reduced footprint is one of the design goals.
- musl libc - a lightweight, fast, simple, and standards-compliant C library
- olibc - another C library optimized for size and performance, derived from Android bionic libc
- Subset Libc Specification - CELF once considered the possibility of creating a subset libc specification. Some companies have also examined the possibility of modularizing glibc, so that parts of it can be made configurable. Preliminary research indicates that this could be a very difficult thing, since glibc has very messy function interdependencies.

Static Linking

If your set of applications is small, sometimes it makes more sense to statically link your applications than to use shared libraries. Shared libraries by default include all symbols (functions and data structures) for the features a library provides. However, when you static link a program to a library, only the symbols that are actually referenced are linked in and included in the program.

Library reduction

It is possible to reduce the size of shared libraries, by eliminating unused symbols.

MontaVista released a tool for library optimization. This tool scans the entire file system, and can rebuild the shared libraries for the system, including only the symbols needed for the set of applications in the indicated file system.

Care needs to be taken with this approach, since it may make it difficult to use add-on programs or or do in-field upgrades (since symbols required by the new software may not be present in the optimized libraries). But for some fixed-function devices, this can reduce your library footprint dramatically.

See http://libraryopt.sourceforge.net/

Deferred Library Loading

It is possible to reduce the RAM runtime footprint for a product, by lazily loading shared libraries, and by breaking up library dependencies. Panasonic did some research into a process called Deferred Library Loading, which they presented at ELC 2007.

See the Deferred Dynamic Loading (pdf) presentation.

Execute-in-place

You can save RAM memory by using some text or data directly from flash.

Kernel XIP

By executing the kernel in-place from flash, it is possible to save RAM space.

- see Kernel XIP

Application XIP

By executing applications in-place from flash, it is possible to save RAM space.

- see Application XIP

Data Read In Place (DRIP)

This is a technique for keeping data in flash, until it is written to, and then making a RAM page for it.

- see Data Read In Place

Size measurement tools and techniques

Kernel size measurement data

- Bloatwatch - a kernel size regression analysis tool.
  - Bloatwatch provides a great amount of detail, and the ability to compare the size of kernel versions over time.

How to measure the kernel image size

- to see the size of the major kernel sections (code and data):

`size vmlinux */built-in.o`

```
[tbird@crest ebony]$ size vmlinux */built-in.o
   text    data     bss     dec     hex filename
2921377  369712  132996 3424085  343f55 vmlinux
 764472   35692   22768  822932   c8e94 drivers/built-in.o
 918344   22364   36824  977532   eea7c fs/built-in.o
  18260    1868    1604   21732    54e4 init/built-in.o
  39960     864     224   41048    a058 ipc/built-in.o
 257292   14656   34516  306464   4ad20 kernel/built-in.o
  34728     156    2280   37164    912c lib/built-in.o
 182312    2704     736  185752   2d598 mm/built-in.o
 620864   20820   26676  668360   a32c8 net/built-in.o
   1912       0       0    1912     778 security/built-in.o
    133       0       0     133      85 usr/built-in.o
```

- to see the size of the largest kernel symbols:
  - `nm --size -r vmlinux`

```
[tbird@crest ebony]$ nm --size -r vmlinux | head -10
00008000 b read_buffers
00004000 b __log_buf
00003100 B ide_hwifs
000024f8 T jffs2_garbage_collect_pass
00002418 T journal_commit_transaction
00002400 b futex_queues
000021a8 t jedec_probe_chip
00002000 b write_buf
00002000 D init_thread_union
00001e6c t tcp_ack
```

How to measure the memory usage at runtime

See Runtime Memory Measurement for a description of ways to measure runtime memory usage in Linux.

Also, see Accurate Memory Measurement for a description of techniques (and patches) which can be used to measure the runtime memory more accurately.

Linux size increase from 2.4 to 2.6

Linux increased in size by between 10% and 30% from version 2.4 to 2.6. This incremental growth in kernel size has been a big concern by forum members.

Please see the Szwg Linux 26Data page for supporting data.

- Size Tunables

GCC Code-Size Benchmarking

CSiBE is a code size benchmark for the GCC compiler. The primary purpose of CSiBE is to monitor the size of the code generated by GCC. In addition, compilation time and code performance measurements are also provided.

CSiBE

Case Studies

"Motorola reduction of system size (presumably for cell phones) using 2.4 Linux"

-
    - MotSizeReduction.ppt - this is a placeholder for this Powerpoint as it was too big to upload to the wiki. (why is this even here? TRB)

uClinux

- Here's an article on uClinux running on cortex-M3s. It has lots of good material on the kernel
    - http://electronicdesign.com/embedded/practical-advice-running-uclinux-cortex-m3m4

Linux on MicroControllers (M3 in this case)

- At ELC 2014, Vitaly Wool described running 2.6.33 Linux on a STMicro STM32F4XX
    - Vitaly's presentation: Spreading the disease: Linux on microcontrollers
    - Device has 256K RAM and 2M flash
    - Kernel and apps were XIP

Reduced-size distribution efforts

Here are some projects aimed at building small-sized systems:

- micro-Yocto (2014)
    - Tom Zanussi has lead an effort in the Yocto Project to produce a minimal kernel for very small embedded systems
    - Here's a presentation by Tom at ELC 2014: microYocto and the Internet of Tiny
    - See https://github.com/tzanussi/meta-galileo/raw/daisy/meta-galileo/README for more information
- http://cgit.openembedded.org/meta-micro/
    - It's maintained by Phil Blundell. It appears pretty successful in reducing size of the resulting image whilst keeping the system fairly functional. It uses uClibc
    - Meta-tiny git repository
- There is a project called Poky-tiny, to produce an extremely stripped-down distribution of embedded Linux using the Yocto project.

  - See https://wiki.yoctoproject.org/wiki/Poky-Tiny
  - Poky-tiny is an effort to build a small-footprint system using Yocto, by Darren Hart

```
meta-tiny is my experimental layer where I'm looking at what we can
build with our existing sources and infrastructure. I've found that we
can cut the image size to about 10% of core-image-minimal without
changes to source code, but dropping a lot of functionality. We can get
to something like 20% while still maintaining ipv4 networking.
```

-
  - Presentation: Tuning Linux For Embedded Systems: When Less Is More

Other Tidbits on System Size

Memory leak detection for the kernel

Catalin Marinas of ARM has been recently (as of 2.6.17?) been posting a memory leak detector for the Linux kernel. It may get mainlined in the future. Here's a link to the LKML discussions around it: http://lkml.org/lkml/2006/6/11/39

How System Size may affect performance

It has long been theorized that reducing system size could provide a performance benefit because it could reduce cache misses. There does not appear to be hard data to support this theory on Linux, but this has been discussed on the kernel mailing list.

See this post by Linus Torvalds

Stripping down the filesystem of a desktop distribution

Here is a good document with tips on how to strip out unneeded files from a desktop distribution. The example distribution used here is Linux From Scratch, but the tips should work with many distributions.

http://www.linuxfromscratch.org/hints/downloads/files/OLD/stripped-down.txt

Extremely-minimal systems

This section lists various efforts to produce the absolute smallest system possible.

- Vitaly Wool describes running 2.6.33 on an ST microcontroler with 2MB flash and 256K ram
  - Linux for Microcontrollers: Spreading the Disease (PDF) (presented at ELC in April 2014)
- Someone is running an old version of BSD on a small processor with only 128K (that's right 'K'!!) of RAM.
  - http://olimex.wordpress.com/2012/04/04/unix-on-pic32-meet-retrobsd-for-duinomite/

Category:

- System Size

> From: eLinux.org

Compiler Optimization

Here's a good overview on compiler optimizations: http://en.wikipedia.org/wiki/Compiler_optimization

Here's some info about GCC optimization techniques: http://www.redhat.com/software/gnupro/technical/gnupro_gcc.html

Effects of optimization options are explained in this LJ article.

A note of warning from Gentoo wiki on optimization flags:

-O3: This is the highest level of optimization possible, and also the riskiest. It will take a longer time to compile your code with this option, and in fact it should not be used system-wide with gcc 4.x. The behavior of gcc has changed significantly since version 3.x. In 3.x, -O3 has been shown to lead to marginally faster execution times over -O2, but this is no longer the case with gcc 4.x. Compiling all your packages with -O3 will result in larger binaries that require more memory, and will significantly increase the odds of compilation failure or unexpected program behavior (including errors). The downsides outweigh the benefits; remember the principle of diminishing returns. Using -O3 is not recommended for gcc 4.x.

In the following e-mail, Jim Wilson, who apparently supports gcc, writes:

```
From: Jim Wilson <wilson at specifixinc dot com>
Date: Thu, 29 Apr 2004 15:58:28 -0700
Subject: Re: optimization issue about -O2 and -Os
------------------------------------------------------------
...
The -Os option is buggy. You might want to report a bug into our bugzilla
bug datase. See http://gcc.gnu.org/bugs.html for more info on reporting bugs.

Though the -Os option is based on the -O2 option, it is a different option, that
generates different code, and has different bugs.
```

Tim Riker: this is a bit overly dramtic. -Os is widely used and widely supported. The link is to a thread about general information and does not refer to any specific bug from what I can see. Try -Os out. If you have issues, try -O2 instead. In general -Os will work. Be very careful in tweaking kernel optimizations. There is kernel code that only works with the existing optimizations.

Gentoo has also a very good overview over Safe Cflags for different architectures and cpus.

Link-time optimization (LTO)

- gcc front-ends (parsers) produce GIMPLE, which is in "static single assignment" (SSA) form
- Then, gcc optimizes the code, and converts to RTL (Register Transfer Language)
- RTL is converted to assembler by an architecture-specific back-end. Then the assembler is called to convert to machine code
- Finally, the linker is called to combine object files

gcc LTO support

- if -flto is used, then LTO information (GIMPLE) is stored in a special ELF section of a .o file, and used at link time to perform more optimization
- You may need to use -fwhole-program in conjunction with -flto at link time in order to get the full set of optimizations
- Using this option requires a lot of memory and takes more time to build the kernel
- Some resources:
  - http://kemiisto.ru/2011/09/gcc-lto-3-basic-usage/
  - See the -flto section of: http://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

Linux kernel LTO support

Andi Kleen produced a set of patches to support LTO in the Linux kernel (originally for version 3.6 of the kernel and gcc 4.7)

- Link-time optimization for the kernel (LWN.net)
- Code is available at: https://github.com/andikleen/linux-misc
    - see the lto-3.x branches
    - note that the code requires the const-sections patches, gcc 4.7 and a special binutils as well, in order to work
        - as of August 2012, this code was considered highly experimental

Categories:

- System Size
- Compiler

> From: eLinux.org

Compressed printk messages

Summary Compressed printk messages

Proposer Tim Bird - Sony Mobile Communications

Status Selected to be sponsored by the CE Workgroup

Contents

Description

Attempts have been made in the past to compress printk messages to save kernel runtime footprint. There is an option to disable all printks, but many embedded developers do not use it, even when they find the space savings attractive, because they still would like to see kernel debug messages.

This project would consist of researching mechanisms that could be used to automatically (at compile-time) compress the kernel's printk messages, and transparently expand them at runtime.

The goal would be to no have user-visible change in behaviour for the kernel, as well as no required developer-visible changes in the source code. Probably, a new kernel option would be used to control this feature.

The work would involve parsing the kernel source code, extracting the messages (and possibly replacing them in source, during the compilation), compressing them, and replacing the original strings with references to the compressed messages (in a way that the messages can be uncompressed transparently at runtime.)

Miscelaneous issues

There may be an issue with finding all kernel messages, due to the number of macros (probably in the hundreds) that are used to wrap printks. This feature might still be useful, even if not all kernel messages could be converted, as long as significant size savings were made available by using the feature. Significant size savings would be on the order of 50K to 100K.

There might also be some benefit from message consolidation. (I don't know if the compiler already coalesces identical strings, but this system should be able to.)

Related work

- Timothy Miller did some work on this in 2003
  - See http://lwn.net/Articles/28935/
  - See https://lkml.org/lkml/2003/6/6/207
- See also some ideas here:
  - http://selenic.com/pipermail/linux-tiny/2005-June/000208.html

Scope

Unknown - 4 to 8 weeks?

Contractor Candidates

None yet.

Comments

This project was proposed in 2012, but not sponsored that year.

Jean-Christophe PLAGNIOL-VILLARD says feature could be useful for barebox as well.

Categories:

- Project proposals 2012
- Project proposals 2013

> From: eLinux.org

Compressed printk messages - Results

Contents

The origin

Inspiration for this project originated in experiments conducted by Timothy Miller in 2003[1]. His experiments lead to the assumption that printk strings in the Linux Kernel can be compressed by 50% (~4% of the total size). No code or patches have been posted, yet the description of his ideas and the published codebook allow a discussion of his approaches, and of general challenges with this topic.

Timothy's approach[2]

1. Compile kernel and keep `.i` -files
2. Filter them for printk strings
3. Compress those strings using tokenization

4. Create copies of the source files
5. There, replace strings with tokens
6. Compile again

Further notes:

- not even sure depacking at printk was ever made
- `allyesconfig` was used for the tests
- based on 2.4.20 and 2.5.68

Very intrusive and large overhead, yet three problems identified

1. Extract printk format strings
2. Compress printk format strings
3. Replace printk format strings

Extracting

Problem: Find all printk-strings

- There are *lots* of functions/defines embedding printk/vprintk_emit
- They are nested in all ways you can think of
- Moving target, there will be more like `<new_subsys>_dev_err, ...`

Option 1: Scan the source files

- needs to know *all* printk-emerging functions
- misses merging of literals at compiler level
- needs to handle all ways of string concatenation in the source files
- this is what was originally done by Timothy Miller

Option 2: Put printk strings to own section

- scales better (only base functions need to be converted)
- compiler does the merging and concatenation
- loses knowledge where strings came from
- needs non-trivial changes to core functions
- experimentally tried out for this research

Extracting: own section

Here is an example how printk was modified to put the string into a seperate section.

Define a wrapper:

```
+#define __printk(fmt, args...) \
+do { \
+   if (__builtin_constant_p(fmt)) { \
+       static const __attribute__((section("__printk"))) \
+           char __f[] = fmt; \
+       printk(__f, ##args); \
+   } else \
+       printk(fmt, ##args); \
+} while (0)
```

And apply it (one example here):

```
 #define pr_emerg(fmt, ...) \
-   printk(KERN_EMERG pr_fmt(fmt), ##__VA_ARGS__)
+   __printk(KERN_EMERG pr_fmt(fmt), ##__VA_ARGS__)
 #define pr_alert(fmt, ...) \
-   printk(KERN_ALERT pr_fmt(fmt), ##__VA_ARGS__)
+   __printk(KERN_ALERT pr_fmt(fmt), ##__VA_ARGS__)
```

Similar approaches have been implemented for `dev_*` and `BUG/WARN`. However, due to various side effects, a full kernel build could not be achieved in the timeframe for this project.

Upstreaming likeliness

However, since the own section approach touches many files close to the core in nasty ways and will most probably cause side-effects, it is extremely unlikely to be accepted upstream. Same goes for the original approach scanning the intermediate files.

Compressing

Algorithms

Requirements

- needs to handle lots of small strings
- should be instantly available (not somewhere in the middle of packed data)
- no significant overhead, both memory and cpu-time

Conclusion

- no sliding window algos (LZ and friends): can't extract slices & lots of bitwise shiftings
- no variable length encoding (Huffman and friends): can't extract slices & lots of bitwise shifting
- no frequency based compression (stats3[3]): needs 2MB RAM & also bitshifting

Codebooks to the rescue?

- tokenization is actually a good option
- BytePairEncoding works, too
- both achieve around 50% of compression
- even today around 4% of the kernel size gained
- estimation based on manually compressed printk sections

However:

- `allyesconfig` is unrealistic for tiny systems
- smaller kernels means smaller pool for codes
- what about modules and their strings?
  - share codebook from the monolithic kernel -> modules are tied to that build
  - modules have their own codebook -> overhead eats gain

Brainstorming:

- hardcode one pre-generated codebook especially suited to kernel printk strings?
- would save second kernel compile, too, if codebook is previously known
- loses some compression ratio since it is not the optimal codebook

Codebooks & UTF8

Tokenization and BPE need unused characters for their symbols which might collide with UTF8 encoding. While UTF8 also has 'illegal encodings' which could be hijacked as compression symbols[4], this is approach is hackish and will also decrease the compression factor by 50% (compression symbols are then 16 bit instead of 8).

Reusing unused ASCII characters as compression symbols is technically sensible here. However, giving up UTF8 cleanliness will probably be not well received upstream.

Replacing

Source code level

In the original implementation, the source code was piped through a filter on the second build of the kernel. Not being able to see what was actually compiled is expected to raise eyebrows when upstreaming.

Own section

Compressing the special printk sections turned out to be rather easy with recent binutils. However, updating all references to the strings is expected to be complex. In this timeframe, it was not possible to research the topic. However, heavily modifying object files as a post-processing step, maybe with modified binutils, is expected to have serious problems upstream.

Conclusion

Summary

All solutions to the problems investigated here turned to be very hackish already at the drafting phase. The likeliness of going upstream is close to zero. They are also no good candidates for keeping them out-of-tree since they tend to be very fragile when it comes to kernel updates.

Further issues (from a higher level)

It is worth noting that printk strings are only a subset of all strings in the kernel. For example, devicetree uses a lot of strings and keeps adding more. They might be easier to tackle since they are largely accessed via `of_*` functions, but this will also add more complexity. To be worth this effort and receive more gain, the problem should be reevaluated at a higher level, considering all strings, maybe even all .rodata.

Other approaches

Turn the viewpoint

From: Managing Gigabytes, Witten/Moffat/Bell, 1st edition, p. 385:

> We find ourselves in the midst of a practically important and intelectually fascinating convergence between the desire for more and better compression and the need to learn about what 'structure' there is in data.

So, trying to understand the structure and improve from there:

- observations from 3.16-rc5:
  - x86-64 allyesconfig
  - arm-cortexa8 customer kernel
- maybe a bit biased for device drivers

Print from central locations

Proposal:

- strings should be emitted from as centralized locations as possible
- bonus: consistent messages

Already applied examples:

- OOM error message removal (mm core will complain anyhow)
- `devm_ioremap_resource()` (unifies error handling for `devm_ioremap` et al.)

Further possibilities:

- have basic functions not printing error messages
- have a convenience function suitable for most cases printing consistent error messages
- `devm_get_optional` are also good candidates

Simple tests:

- Removing error strings for `devm_clk_get` saved 20K instantly
- lots of other candidates

Prefixes

- usually done by a literal prefixing the format string
- creates unique strings which have redundancy in them
- use `dev_*` and friends if possible

Prefixes: Dead simple tinyfication

How `pr_fmt` gets applied:

```
#define pr_alert(fmt, ...) \
    printk(KERN_ALERT pr_fmt(fmt), ##__VA_ARGS__)
```

How to simplify it:

```
-#define pr_fmt(fmt) KBUILD_MODNAME ": " fmt
+#define pr_fmt(fmt) "%s" fmt, KBUILD_MODNAME ": "
```

That saved around 15% (or 250 byte) for sn9c20x. Applied to all 900 instances in the kernel, it saved about 30K (or 0.4%). It works better in some places than in others.

Prefixes: More easy stuff

Redefine subsystem printouts:

```
 /* UBI error messages */
-#define ubi_err(fmt, ...) pr_err("UBI error: %s: " fmt "\n",      \
-                 __func__, ##__VA_ARGS__)
+#define ubi_err(fmt, ...) printk("%s%s: " fmt "\n",      \
+                KERN_ERR "UBI error: ",  __func__, ##__VA_ARGS__)
```

That saved around 15% (or 2.5K). UBIFS, JFFS2, SCSI layer seem also to be promising candidates.

Copy'n'paste

Some code duplicates strings too easy[5]:

```
switch (sd->sensor) {
case SENSOR_OV9650:
    ov9650_init_sensor(gspca_dev);
    if (gspca_dev->usb_err < 0)
        break;
    pr_info("OV9650 sensor detected\n");
    break;
case SENSOR_OV9655:
    ov9655_init_sensor(gspca_dev);
    if (gspca_dev->usb_err < 0)
        break;
    pr_info("OV9655 sensor detected\n");
    break;
case SENSOR_SOI968:
    soi968_init_sensor(gspca_dev);
    if (gspca_dev->usb_err < 0)
        break;
    pr_info("SOI968 sensor detected\n");
    break;

/* ... 7 more ... */
```

And in the init functions:

```
if (gspca_dev->usb_err < 0)
    pr_err("OV9650 sensor initialization failed\n");
...
if (gspca_dev->usb_err < 0)
    pr_err("OV9655 sensor initialization failed\n");
...
if (gspca_dev->usb_err < 0)
    pr_err("SOI968 sensor initialization failed\n");
...
```

A little love helps a lot here \<3

- proper cleanups are most sustainable
- will not only remove strings but code as well

For example, here:

- refactor init routines to return error values
- print detected/failed messages depending on error value

- use a table for sensor names and keep the rest of the string static

Layer 8

Be aware when adding strings:

- be precise with printk level (error, warning, debug). This makes compiling out based on level more useful.
- be conservative with non debug strings. We need rules of thumb here...
- try to be consistent with the strings you create.
- generate strings instead of copy-pasting similar strings
- try to avoid prefixes; especially for drivers, there are plenty of alternatives
- strings cost (a little), so they should be worth it

Summary

Compressed printk-strings, only for corner-cases

- the price (overhead, complexity) for compressed printk-strings is still high
- no existing implementation, depends heavily on use case (e.g. how are modules handled?)
- the gain can be expected to be still around 4%
  - note that you will lose compression rate on the compressed kernel image because the already compressed printk data will compress worse
- lots of side effects and huge increase of build time
- looking at printk-strings is not enough (e.g. DT bindings)

General improvements first, benefits for all

- there is quite some potential to simply reduce number of strings, especially through centralization
- ...which mostly make strings more consistent, too
- cleaning up sloppy code is good, too
- should be the first goal before stepping further, lots of KB can be saved
- then have another look for patterns

- ↑ http://elinux.org/Compressed_printk_messages

- ↑ http://lwn.net/Articles/28935/
- ↑ http://www.scirp.org/journal/PaperInformation.aspx?PaperID=40783
- ↑ http://en.wikipedia.org/wiki/UTF-8#Codepage_layout
- ↑ https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/drivers/media/usb/gspca/sn9c20x.c?id=v3.16-rc5#n1813

> From: eLinux.org

Data Read In Place

This page has information about "Data Read-In-Place", which is of interest to CE Linux Forum members, because it allows data pages to reside on ROM or flash, until they are written to. This is essentially a form of XIP, or copy-on-write, for data pages. XIP is used to keep text segment pages in flash permanently. This technique ("DRIP") is used to keep data pages in flash until they are written to. Since many application data pages are never written to, the net effect is a substantial reduction in RAM usage for application data segments. This feature was also called "Data Allocate On Write" previously, but the name "Data Read In Place" is closer to the already-existing term for text (Execute In Place), and is now preferred.

The total effect for one system measured by Panasonic was a reduction of 26% of the page cache allocated to processes, when the product was in the stand-by state.

The technique was described by Masashige Mizuyama, Chief Architect in the System Architecture Develompent Group, Base System Development Center, Panasonic Mobile Communications Co., Ltd.

Description

There is no need to change kernel code for this feature. We changed the dynamic linker (in glibc of MVL CEE3.1) only. This was used with a 2.4.x Linux kernel.

Usually, the dynamic linker maps each ELF segment to the virtual address space of the process, using mmap.

We change it as follows:

```
if(segment includes a .data section){
   Do mmap(), forcing PROT_WRITE bit off. ------(1)
   Set PROT_WRITE bit on, with mprotect(). ------(2)
} else {
  Do mmap() as usual.
}
```

This is very simple.

Below is the description of how it works.

When the XIP ELF shared library is dynamically linked at runtime, because the PROT_WRITE bit is off ((1) above) when the section is mmap'ed, the kernel assumes the linker is mapping an XIP text segment. So the kernel builds a page directory table to map every physical ROM page of the segment to the process virtual address space. Each page table entry (PTE) is write-proctected.

Then, becaused of the mprotect call setting PROT_WRITE on the mapped area, the virtual memory area for the segment has write permission (in the kernel vm_area_struct). The write permission combination of PTE and vm_area_struct is identical with a page which is enabled "copy on write".

So, the the pages in the segment are mapped to ROM pages directly until they are written.

This is a kind of "fake" approach to support the feature with minimal changes. So there are some pitfalls to this approach. One problem we already notice is that get_user_pages() does not work a segment mapped like this.

The get_user_pages() function is used for mlock, ptrace and core dump by kernel. So they don't work for the segment correctly with the current implementation.

However, the advantage was much enough for us, we decided to use it. I think the implementation needs to be cleaned up by adding direct kernel support for this type of page mapping. }}}

Documents

- Information about this technology is included in the Panasoinc presentation made at the CELF International Technical Jamboree.
  - Making Mobile Phone with CE Linux by Masashige Mizuyama (Panasonic Mobile Communications), June 2005
    - see pages 6 - 12 of the presentation

draft patch

This patch can be applied to the runtime linker [what program is this? ld-linux.so??]

```
*** dl-load7.c    Mon Jul 11 21:26:47 2005
--- dl-load.c    Sat Jan  8 11:37:38 2005
***************
*** 801,819 ****
--- 801,849 ----
    if (! (locked_load_mode & (RTLD_LOCK_DEPENDENT_LIB_PAGES
                   | RTLD_LOCK_LIB_PAGES)))
      {
+      if((prot & PROT_WRITE) != 0 ){
+     prot = (prot & ~PROT_WRITE);
+       mapat = __mmap ((caddr_t) mapstart, len, prot,
+               fixed|MAP_COPY|MAP_FILE,
+               fd, offset);
+         if (mapat != MAP_FAILED){
+       prot = (prot | PROT_WRITE);
+       if( __mprotect(mapat,len,prot)==-1){
+         return N_("failed to map segment from shared object");
+       }
+     } else {
+       return N_("failed to map segment from shared object");
+     }
+      } else {
        mapat = __mmap ((caddr_t) mapstart, len, prot,
                fixed|MAP_COPY|MAP_FILE,
                fd, offset);
        if (mapat == MAP_FAILED)
      return N_("failed to map segment from shared object");
+      }
      }
    else if (locked_load_mode & RTLD_LOCK_MLOCK)
      {
+      if((prot & PROT_WRITE) != 0 ){
+     prot = (prot & ~PROT_WRITE);
+       mapat = __mmap ((caddr_t) mapstart, len, prot,
+               fixed|MAP_COPY|MAP_FILE,
+               fd, offset);
+         if (mapat != MAP_FAILED){
+       prot = (prot | PROT_WRITE);
+       if( __mprotect(mapat,len,prot)==-1){
+         return N_("failed to map segment from shared object");
+       }
+     } else {
+       return N_("failed to map segment from shared object");
+     }
+       } else {
        mapat = __mmap ((caddr_t) mapstart, len, prot,
                fixed|MAP_COPY|MAP_FILE,
                fd, offset);
        if (mapat == MAP_FAILED)
      return N_("failed to map segment from shared object");
+       }
        if (mlock((caddr_t) mapat, len) != 0)
      {
        return N_("failed to memory lock segment from shared object");
```

Category:

- System Size

> From: eLinux.org

Kernel Size Reduction Work

This page has a list of of recent (as of 2014) ideas and projects for Linux kernel size reduction.

As of June, 2014, Linux Tiny was no longer maintained. However, new efforts to support Linux as a viable option on micro-controllers and deeply embedded systems have led to renewed interest in extreme size reduction of the kernel. Some good projects were described at ELC 2014.

Contents

Recent work

- Tom Zanussi and the Micro-yocto project (2014)
  - Micro-Yocto is an effort in the Yocto Project to produce a minimal kernel for very small embedded systems
  - microYocto and the Internet of Tiny (PDF)
    - Presentation by Tom Zanussi at ELC 2014
  - See https://github.com/tzanussi/meta-galileo/raw/daisy/meta-galileo/README for more information
- Vitaly Wool described running 2.6.33 on an ST microcontroler with 2MB flash and 256K ram (2014)
  - Linux for Microcontrollers: Spreading the Disease (PDF) - presented at ELC in April 2014

List of recently used techniques

Vitaly Wool's project

- kernel XIP
- dietnet
- ARM thumb mode
- application XIP on squashfs

micro-yocto project

- net-diet
- LTO
- SYSFS_SYSCALL
- USELIB

- BUG_ON
- X86_IOPORT
- CONFIG_PTRACE
- CONFIG_SIGNALS
- no sys_sendfile

- network reductions:

  - replace kernel stack with user-space stack (LWIP)
  - TCP/UDP echo (requires app changes)

instrumentation

microYocto tracing hash triggers

auto-reduce

- LTO
- syscall filtering
- command line filtering
- global constraints
- kernel stack size

Ideas for new reductions

- trace-guided optimization
- cold code compression
  - use U of Gent code to re-try cold code compression

People or groups interested in size reductions

individuals

- Josh Triplett - e-mail:josh at joshtriplett dot org
  - new linux tiny patches
- Tom Zanussi - e-mail: tom dot zanussi at linux dot intel dot com
  - microYocto
- Andi Kleen - e-mail: ak at linux dot intel dot com
  - netdiet patches
- Alan Cox - email: alan dot cox at linux dot intel dot com
- Shinsuke Kato - e-mail: kato dot shinsuke at jp dot panasonic dot com
- Tim Bird - e-mail: tim dot bird at sonymobile dot com
  - auto-reduce stuff
- Vitaly Wool - e-mail: vitaly dot wool at softprise dot net
  - extreme microcontroller Linux (\<256K RAM)
- Phil Blundell - e-mail:
  - meta-micro (OE-based small distro - last work in 2012)

companies

- Emcraft Systems sells a variety of microcontroller-based boards and products

Meetings and Discussions

Kernel Summit 2014

Josh Triplett has proposed a session discussing size issues at the 2014 Kernel Summit. His draft list of proposed topics was the following: Topics:

- An overview of why the kernel's size still matters today ("but don't we all have tons of memory and storage?")
- Tiny in RAM versus tiny on storage.
- How much the kernel has grown over time.
- How size regressions happen and how to avoid them
- Size measurement, bloat-o-meter, allnoconfig, and other tools
- Compression and the decompression stub
- Kconfig, and avoiding excessive configurability in the pursuit of tiny
- Optimizing a kernel for its exact target userspace.
- Examples of shrinking the kernel
- Discussion on proposed ways to make the kernel tiny, how much they might save, how much work they'd require, and how to implement them with minimal impact to the un-shrunken common case.

(see http://lists.linuxfoundation.org/pipermail/ksummit-discuss/2014-May/000001.html for the discussion thread)

Size/IOT summit at ELC Europe 2014

We are considering holding a Size/IOT meeting at ELC Europe. Details will be placed here should this meeting end up being organized.

Category:

- System Size

> From: eLinux.org

Kernel Size Tuning Guide

This document describes how to configure the Linux kernel to use a small amount of memory and flash.

⚠ *Note: This document is a work in progress. Please feel free to add material anywhere you have additional information or data. Sections of this document which need additional work are denoted with [FIXTHIS] markers.*

Contents

Introduction

One big problem area when using Linux in an embedded project is the size of the Linux kernel.

Measuring the kernel

There are 3 aspects of kernel size which are important:

the size of the kernel image stored in flash (or other persistent storage)

the static size of kernel image in RAM (usually, this will be the size of the uncompressed image)

- This includes the text, data, and BSS segments of the kernel at the time it is loaded. The text and BSS segments will stay the same size for the kernel throughout it execution. However, the data and stack segments may grow according

to the needs of the system.

the amount of dynamic RAM used by the kernel.

- This will fluctuate during system execution. However, there is a baseline amount of memory which is allocated at system startup. Application-specific RAM can be calculated to be above this minimal amount of required RAM.

For now, this document ignores Execute-In-Place (XIP) and Data-Read-In-Place (DRIP) techniques, the use of which have an impact on the amount of flash and RAM used by the kernel. See the following online resources for more information about these techniques: Kernel XIP and Data Read In Place

Measuring the kernel image size

The compressed kernel image is what is stored in the flash or ROM of the target device. The size of this image can be obtained by examining the size of the image file in the host filesystem with the '`ls -l`' command:

- for example: '`ls -l vmlinuz`' or '`ls -l bzImage`' (or whatever the compressed image name is for your platform.)

Measuring the kernel text, data and bss segments

Use the `size` command to determine the size of the text, data, and BSS segments of a kernel image.

Note that the BSS segment is not stored in the kernel image because it can be synthesized at boot time by filling a block of memory with zeros. Note also that portions of the kernel text and data are set aside in special initialization segments, which are discarded when the kernel finishes booting. Because of these factors, the size command does not give you an exactly correct value for the static kernel RAM size. However, it can be used as a reasonable estimate.

To use the size command, run it with the filename of the uncompressed kernel image (which is usually `vmlinux`).

- for example: '`size vmlinux`'

Example output:

```
   text    data     bss     dec     hex filename
2921377  369712  132996 3424085  343f55 vmlinux
```

Measuring and comparing sub-parts of the kernel

In order to find areas where the kernel size can be reduced, it is often useful to break down the static size of the kernel by sub-system or by kernel symbol. The following sections describe how to see the size of each kernel sub-system, how to see the size of individual kernel symbols, and how to compare the size of symbols between two kernel versions. This is useful because as you make changes to the kernel configuration you can determine what part of the kernel is affected by the change. From this information you may be able to predict what the affect of the change will be, and decide whether the change is acceptable.

Measuring major kernel subsystems

The major sub-systems of the kernel are put into library object files named `built-in.o` in the corresponding sub-directory for that sub-system within the kernel build directory. The major sub-directories, at the time of this writing (for kernel 2.6.17) are: `init, user, kernel, mm, fs, ipc, security, crypto, block, ltt, drivers, sound, net, lib`

To see the size of the major kernel sections (code, data, and BSS), use the `size` command, with a wildcard for the first level of sub-directory:

- `size */built-in.o`

You can pipe this output through `sort` to sort by the largest libraries:

- `size */built-in.o | sort -n -r -k 4`

Example output:

```
731596   53144   33588  818328   c7c98 drivers/built-in.o
687960   24972    2648  715580   aeb3c fs/built-in.o
547844   19508   28052  595404   915cc net/built-in.o
184072    6256   32440  222768   36630 kernel/built-in.o
141956    3300    2852  148108   2428c mm/built-in.o
 68048    1804    1096   70948   11524 block/built-in.o
 26216     768       0   26984    6968 crypto/built-in.o
 17744    2412    2124   22280    5708 init/built-in.o
 20780     292     124   21196    52cc ipc/built-in.o
 18768      68       0   18836    4994 lib/built-in.o
  2116       0       0    2116     844 security/built-in.o
   134       0       0     134      86 usr/built-in.o
  text    data     bss     dec     hex filename
```

To see even greater detail, you can examine the size of `built-in.o` files even deeper in the kernel build hierarchy, using the `find` command:

- `find . -name "built-in.o" | xargs size | sort -n -r -k 4`

Example output:

```
731596   53144   33588  818328   c7c98 ./drivers/built-in.o
687960   24972    2648  715580   aeb3c ./fs/built-in.o
547844   19508   28052  595404   915cc ./net/built-in.o
260019    9824    4944  274787   43163 ./net/ipv4/built-in.o
184072    6256   32440  222768   36630 ./kernel/built-in.o
...
```

 **Note: Please be careful interpreting the results from the size of the `built-in.o` files in sub-directories. In general, the object files are aggregated into the libraries of parent directories, meaning that many object files will have their size counted twice. You cannot simply add the columns for an indication of the total kernel size**

Measuring individual kernel symbols

You can measure the size of individual kernel symbols using the 'nm' command. Using the `nm --size -r vmlinux`

```
[tbird@crest ebony]$ nm --size -r vmlinux | head -10
00008000 b read_buffers
00004000 b __log_buf
00003100 B ide_hwifs
000024f8 T jffs2_garbage_collect_pass
00002418 T journal_commit_transaction
00002400 b futex_queues
000021a8 t jedec_probe_chip
00002000 b write_buf
00002000 D init_thread_union
00001e6c t tcp_ack
```

Legend: The columns of this output are:

1. size in bytes (in hexadecimal)
2. symbol type
3. symbol name.

The symbol type is usually one of:

- 'b' or 'B' for a symbol in the BSS segment (uninitialized data),
- 't' or 'T' for a symbol in the text segment (code), or
- 'd' or 'D' for a symbol in the data segment.

Use ' `man nm` ' for additional information on the ' `nm` ' command.

Comparing kernel symbols between two kernel images

Use the bloat-o-meter command, found in the kernel source `scripts` directory, to compare the symbol sizes between two kernel images.

- `\ /scripts/bloat-o-meter vmlinux.default vmlinux.altconfig`

If you get an error: 'chmod a+x \/scripts/bloat-o-meter'

Example output, comparing a baseline kernel to one configured with CONFIG_PRINTK=n:

```
[] $ ../../linux/scripts/bloat-o-meter vmlinux.baseline vmlinux.no-printk
add/remove: 5/23 grow/shrink: 8/1541 up/down: 1141/-199824 (-198683)
function                                     old     new   delta
proc_ioctl_default                             -     610    +610
proc_reapurb                                   -     296    +296
proc_disconnectsignal                          -      88     +88
proc_releaseinterface                          -      72     +72
proc_claiminterface                            -      36     +36
xprt_adjust_cwnd                             169     182     +13
do_timer                                    1052    1063     +11
i8042_controller_reset                        78      84      +6
serio_init                                   167     172      +5
usb_exit                                      80      81      +1
early_uart_console_init                       45      46      +1
console_unblank                              103     104      +1
console_conditional_schedule                  21      22      +1
parse_early_param                            102     101      -1
machine_emergency_restart                    249     248      -1
console_callback                             239     238      -1
arch_align_stack                              45      44      -1
quirk_p64h2_1k_io                            183     181      -2
printk_time                                    4       -      -4
printk_cpu                                     4       -      -4
oops_timestamp.7                               4       -      -4
neigh_resolve_output                         733     729      -4
msg_level.4                                    4       -      -4
...
de_dump_status                              1586     313   -1273
decode_getfattr                             3156    1748   -1408
ext3_fill_super                             5980    4545   -1435
usbdev_ioctl                                6476    4846   -1630
usb_get_configuration                       4001    1878   -2123
proc_submiturb                              2294       -   -2294
__log_buf                                 131072       - -131072
```

Kernel Size Tuning features

The Linux kernel includes a number of options for to control the features and options it supports. The kernel, over time, has accumulated a large set of features and capabilities. But many features are not needed in Consumer Electronics products. By carefully tuning the kernel options, you can omit many parts of the kernel and save memory in your product.

Linux-tiny patches

The Linux-tiny patch set is a set of patches maintained by Matt Mackall developed with the intent to help a developer reduce the size of the Linux kernel.

These patches are described at: Linux Tiny

The Linux-tiny patch set includes a number of different patches to allow the kernel to be reduced in size. Sometimes, the size reductions are accomplished by reducing the number of objects for a particular features (like the number of possible swap areas, or the number of tty discipline structures). Sometimes, the size reductions are achieved by removing features or functions from the kernel.

Here is a list of the individual Linux-tiny patches that are available for the 2.6.22 kernel at Linux Tiny Patch Details

Please note that the last patch in this list ("do-printk") is available separately from the main Linux-tiny patch set. Please find this patch at: Do Printk

The patches listed in this table represent patches that can be applied to a 2.6.16 Linux kernel. However, as of version 2.6.16, many options for reducing the kernel were already available in Linux. A list of options, both from these patches and from existing code, which are interesting for tuning the kernel size is provided in the section: "Kernel configuration Options"

How to configure the kernel

[FIXTHIS - need detailed kernel configuration instructions]

- use 'make menuconfig'
- perform thorough testing of your library and applications with the smaller config
- development vs. deployment configurations
- describe all_no config - most times it won't boot.

Kernel Configuration Options

Here is a table of kernel configuration options, including a description, the default value for a kernel, and the recommended value for a smaller configuration of the kernel:

CONFIG option	Description	Default	Small
CONFIG_CORE_SMALL	tune some kernel data sizes	N	Y
CONFIG_NET_SMALL	tune some net-related data sizes	N	Y
CONFIG_KMALLOC_ACCOUNTING	turn on kmalloc accounting	N	Y *
CONFIG_AUDIT_BOOTMEM	print out all bootmem allocations	N	Y *
CONFIG_DEPRECATE_INLINES	cause compiler to emit info about inlines	N	Y *
CONFIG_PRINTK	allow disable of printk code and message data	Y	N
CONFIG_BUG	allow elimination of BUG (and BUG_ON??) code	Y	N
CONFIG_ELF_CORE	allow disabling of ELF core dumps	Y	N
CONFIG_PROC_KCORE	allow disabling of /proc/kcore	Y	N
CONFIG_AIO	allow disabling of async IO syscalls	Y	N
CONFIG_XATTR	allow disabling of xattr syscalls	Y	N
CONFIG_FILE_LOCKING	allow disabling of file locking syscalls	Y	N
CONFIG_DIRECTIO	allow disabling of direct IO support	Y	N
CONFIG_MAX_SWAPFILES_SHIFT	number of swapfiles	5	0
CONFIG_NR_LDISCS	number of tty line disciplines	16	2
CONFIG_MAX_USER_RT_PRIO	number of RT priority levels (schedule slots)	100	5
Other config options	These are not in Linux-tiny, but help with size	default	small
CONFIG_KALLSYMS	load all symbols for debugging/kksymoops	Y	N
CONFIG_SHMEM	allow disabling of shmem filesystem	Y	N +
CONFIG_SWAP	allow disabling of support for a swap segment (virtual memory)	Y	N
CONFIG_SYSV_IPC	allow disabling of support for System V IPC	Y	N +
CONFIG_POSIX_MQUEUE	allow disabling of POSIX message queue support	Y	N +

CONFIG_SYSCTL	allow disabling of sysctl support	Y	N +
CONFIG_LOG_BUF_SHIFT	control size of kernel printk buffer	14	11
CONFIG_UID16	allow support for 16-bit uids	Y	??
CONFIG_CC_OPTIMIZE_FOR_SIZE	Use gcc -os to optimize for size	Y	Y
CONFIG_MODULES	allow support for kernel loadable modules	Y	N +
CONFIG_KMOD	allow support for automatic kernel module loading	Y	N
CONFIG_PCI	allow support for PCI bus and devices	Y	Y -
CONFIG_XIP_KERNEL	allow support for kernel Execute-in-Place	N	N
CONFIG_MAX_RESERVE_AREA	??	??	??
CONFIG_BLK_DEV_LOOP	support for loopback block device	Y	Y -
CONFIG_BLK_DEV_RAM	support for block devices for RAM filesystems	Y	Y -
CONFIG_BLK_DEV_RAM_COUNT	Number of block devices for RAM filesystems	16	2?
CONFIG_BLK_DEV_RAM_SIZE	Size of block device struct for RAM filesystems	4096	??
CONFIG_IOSCHED_AS	Include Anticipatory IO scheduler	Y	Y
CONFIG_IOSCHED_DEADLINE	Include Deadline IO scheduler	Y	N +
CONFIG_IOSCHED_CFQ	Include CFQ IO scheduler	Y	N +
CONFIG_IP_PNP	support for IP autoconfiguration	Y	N +
CONFIG_IP_PNP_DHCP	support for IP autoconfiguration via DHCP	Y	N +
CONFIG_IDE	support for IDE devices	Y	N +
CONFIG_SCSI	support for SCSI devices	Y	N +

Legend:

- "Y *" - Set to 'Y' for measurement during development, and set to 'N' for deployment.
- "N +" - Whether you can set this to 'N' depends on whether this feaure is needed by your applications.
- "Y -" - You probably need this, but it might we worth checking to see if you don't.

Special Instructions for some kernel options

How to use CONFIG_PRINTK

If the "do-printk" patch is applied, there are two options which control the compilation of printk elements in the kernel: CONFIG_PRINTK_FUNC and CONFIG_PRINTK. You can use these options to control how much printk support the kernel provides, and to control on a global basis whether any printk messages at all are compiled into the kernel. Another special preprocessor variable is also available, called DO_PRINTK, which provides the ability to enable printk messages inside a single C compilation unit, even if printk messages are disabled globally.

This section explains how to use these features to reduce the kernel size, while still enabling sufficient printk messages to be useful during development and deployment.

The CONFIG_PRINTK option disables all of the kernel printk calls. By setting this option to 'N' in your kernel configuration, all uses of "printk" throughout the kernel source are turned into empty statements, and omitted when the program is compiled. This provides a substantial size savings, since the kernel messages often account for more than 100 kilobytes of space in the kernel image. Setting this option to 'N' will not, however, remove the actual

```
printk
```

code itself (just the calls to

```
printk
```

). The CONFIG_PRINTK_FUNC option controls whether the

```
printk
```

function and various helper functions are compiled into the Linux kernel. When this is set to 'N', CONFIG_PRINTK is automatically set to 'N', and no printk messages are compiled into the kernel. This usually saves about another 4K of size in the kernel image.

By using both CONFIG_PRINTK and CONFIG_PRINTK_FUNC, you can reduce the size of the kernel image (and that flash and RAM it requires). However, there is a drawback to eliminating all the messages. Obviously, it is then not possible to get any status, diagnostic or debug messages from the kernel! Another mechanism is available, which allows you to control on a per-file basis which printk calls are compiled into the kernel. This is the pre-processor variable DO_PRINTK.

To use DO_PRINTK, set CONFIG_PRINTK to 'N' and CONFIG_PRINTK_FUNC to 'Y' in your kernel configuration. This will globally disable all printk calls in the kernel. Now, determine the C files where you wish to enable printk messages, and add the line:

```
#define DO_PRINTK 1
```

at the top of each file. Now, the printk calls in those files will be compiled normally. Printk calls in other modules will be omitted.

⚠ - **Important Note:** The DO_PRINTK variable controls how the preprocessor will treat printk statements in the code. For this reason, this statement **MUST** appear at the top of the file, before any

```
#include
```

lines. In order to change the set of printk messages preserved in the code, you will need to modify the

```
DO_PRINTK
```

lines, and recompile the kernel. (There is no runtime control of the printk calls.) This is a simple mechanism, but it does provide a way to omit **most** of the printk messages from the kernel while still preserving some messages that may be useful during

development or on a deployed product.

In review, there are basically 3 different settings combinations for CONFIG_PRINTK_FUNC and CONFIG_PRINTK that make sense:

Settings		Explanation
CONFIG_PRINTK_FUNC	CONFIG_PRINTK	
Y	Y	This is the default setting for the kernel configuration. In this setting the `printk` code is compiled into the kernel, and all printk calls throughout the entire source code are also compiled as part of the kernel.
Y	N	This leaves the actual printk() routine in the kernel, but disables all calls to printk throughout the entire source code. However, you can use DO_PRINTK in individual modules to enable the printk calls from those modules.
N	N	This removes the printk() routine from the kernel, and disables all kernel printk messages, and gives the smallest kernel code and data size. DO_PRINTK will NOT enable any module-specific printk calls.

Booting without SysFS

(copied from linux-tiny wiki)

Turning off sysfs support can save a substantial amount of memory in some setups. One big downside is that it breaks the normal boot process because the kernel can no longer mapa symbolic device name to the internal device numbers.

Thus, you will need to pass a numeric device number in hex. For example, to boot off /dev/hda1, which has major number 3 and minor 1, you'll need to append a root== option like this:

```
/boot/vmlinuz root==0x0301 ro
```

Booting without /proc fs

It is also possible to boot with

```
/proc
```

fs, but many programs expect this psuedo-filesystem to be present and mounted. For example,

```
free
```

and

```
ps
```

are two commands which retrieve information from

```
/proc
```

in order to run.

list some workarounds here

Using kernel memory measurement features

⚠ FIXTHIS - need instruction on bootmem auditing and counting inlines - need more detail for kmalloc accounting

Kmalloc Accounting

This is a features of Linux-tiny, which tracks callers of kmalloc and kfree, and produces summary statistics for kernel memory allocations, as well as detailed information about specific kmalloc callers.

This was first published by Matt Mackall in February of 2005, but was not mainlined at that time.

To see results for kernel allocations, follow these steps:

- turn on the CONFIG_KMALLOC option. This will show up on the kernel configuration menus as "Enabled accounting of kmalloc/kfree allocations?"
- recompile your kernel
- boot the kernel
- periodically, examine the accounting stats
  - cat /proc/kmalloc

See http://lwn.net/Articles/124374/

Bootmem Auditing

Counting Inlines

Outline

⚠ FIXTHIS - need to review outline and fill in missing material

- Tuning the kernel
- how to measure kernel size
- in-kernel size reporting - kmalloc accounting
- bloat-o-meter
- kernel configuration options
- mainline options
- optional features
- minimal config
- sufficient API?
- POSIX compliance
- LSB compliance
- LTP compliance
- file systems
- comparison of file system sizes
- compiler options for reducing size
- gcc -os
- gcc -whole-program
- online resources:
- bloatwatch
- kconfigsize

References

- Linux-tiny project web site: [linux-tiny]
- eLinux wiki Linux-tiny page: Linux Tiny
- Matt Mackall's http://elinux.org/images//8/83/Pdf.gif Linux-tiny presentation http://elinux.org/images/d/da/Info_circle.png
- CE Linux Forum resources for reducing system size: System Size

Appendices

Appendix A - Sample minimum configuration for ARM

[FIXTHIS - need ARM minimum config.]

Appendix B - Configuration Option Details

⚠Want to fill in this section with details about configuration options.

For each option, would like to document:

- what is size affect for different option values
- This page & Kernel Size Tuning Guide Config Option Impact describe kernel size and RAM usage impact affected by each configuration option listed in "Kernel Configuration Options" above, on i386.
- what is affect of performance, functionality, etc.
- what programs (if any) will stop working if option is turned off (or reduced)

Appendix C - Things to research

- miniconfigs
- how to use an initramfs (to avoid using NFS-mounted rootfs)
- how to use a local fs (to avoid using NFS-mounted rootfs)
- Eric Biederman's turning off CONFIG_BLOCK - will any FS work after this??
- he got a 2.6.1 kernel (presumably all_no) to: "191K bzImage and a 323K text segment". See here.
- why is networking so big??
- why are file systems so big??
- capture serial output from kernel for size measurement (see grabserial program)

Categories:

- Kernel Size Tuning Guide
- System Size

> From: eLinux.org

Kernel Small Stacks

Here is some random information about small kernel stack sizes.

The default stack size for a process running in kernel space is 8K (as of 2011).

There used to be an option on x86 to reduce the stack size to 4K. And indeed there were efforts in 2006 to make this the default stack size. However, using a small stack opens up the dangerous possibility that the stack will overflow, causing a kernel hang.

Besides wasting memory, if the stack space is not really needed, 8K stacks also have an effect on, and are affected by, general kernel memory allocation. To create an 8K stack requires an order-1 allocation, meaning that 2 contiguous physical pages must be allocated together in order to create a new process stack. If memory has become fragmented, it may be impossible to fulfill an order-1 allocation, even though individual pages of physical memory may be free. Thus 4K stack allocations (order-0 allocations) are more likely to succeed. This is important for systems operating under extreme memory pressure.

There were years of debate on the kernel mailing list about whether 4K stacks should be the default, and lots of bug reports that ended up being caused stack overflows with 4K stacks enabled. The option to support 4k stacks on x86 was removed in 2010, from kernel version 2.6.37 with this commit: dcfa726280116dd31adad37da940f542663567d0

Contents

Historical information on 4K Stacks

Here are some articles and links to information about 4K stacks:

- 4K Stacks in 2.6 - LWN.net, May 2004
- 4K stacks for everyone? - LWN.net, Sep 2005
- 4K stacks - again - LWN.net, Nov 2005
- 4K stacks by default? - LWN.net, Apr 2008

- http://www.redhat.com/archives/rhl-list/2007-March/msg00854.html

- James Wilkinson message about memory fragmentation.

Stack layout

The kernel stack is laid out with the stack pointer at the top of each stack (at the highest stack address), growing downward for each function call and stack allocation. The thread_info structure for a process is at the bottom of the stack. There is no physical mechanism to detect, at allocation time, if the stack pointer wanders into the thread_info area of the stack. Hence, if the stack overflows (the stack pointer goes into the thread_info area), the behavior of the system is undefined.

Stack structure:

```
top     +----------------+
        | return vals    |
        |   & local vars |
        | ...            |
        |                |
        |                |
        | 0's            |
        | thread_info    |
bottom  +----------------+
```

Stack measuring/monitoring mechanisms

Because of previous efforts to conserve stack space, there are actually a few different mechanisms for monitoring the kernel stack usage. Some tools report on the static size of stack usage by kernel functions (a check which is done by either the compiler or a separate tool operating on the kernel binary), and some mechanisms can report on actual stack utilization at runtime.

CONFIG_FRAME_WARN

This kernel configuration option passes an option to the compiler to cause it to emit a warning when a static stack size for a routine is detected that is larger than the specified threshold. It requires gcc version 4.4 or later in order to work.

The gcc option used is "-Wframe-larger-than=xxx".

By default, CONFIG_FRAME_WARN has the value of 1024, but you can set it to any value from 0 to 8192.

Here is a sample of build output with this option configured to 256:

```
  ...
  CC      ipc/msg.o
  CC      ipc/sem.o
 .../linux-3.0.y/ipc/sem.c: In function 'semctl_main.clone.7':
 .../linux-3.0.y/ipc/sem.c:1021:1: warning: the frame size of 520 bytes is larger than 256 bytes
 .../linux-3.0.y/ipc/sem.c: In function 'sys_semtimedop':
 .../linux-3.0.y/ipc/sem.c:1514:1: warning: the frame size of 472 bytes is larger than 256 bytes
  CC      ipc/shm.o
  CC      ipc/ipcns_notifier.o
```

checkstack.pl

The kernel source includes a script to perform static stack analysis called scripts/checkstack.pl.

Usage is as follows:

```
$(CROSS_COMPILE)objdump -d vmlinux | scripts/checkstack.pl [arch]
```

Replace [arch] with the architecture of the kernel being analyzed. Several architectures are supported, including arm, mips and x86. You should use a cross-objdump that matches the architecture you compiled the kernel for. For example, if you used: arm-gnueabi-linux-gcc as your compiler, you would use arm-gnueabi-linux-objdump as your object dump program. This should have been included in your cross-compiler toolchain package.

Below is some sample output from using checkstack.pl. Note that the file is first dumped to an assembly file (.S), and then piped to checkstack.pl. You can examine the assembly file to see in detail the instructions used to reserve space on the stack, for routines of interest found by checkstack.pl.

An item in brackets is a module name, in case of a loadable module. The number at end is stack depth detected for function. The Leading value is the address of the stack reservation code.

```
$ arm-eabi-objdummp -d vmlinux -o vmlinux-arm.S
$ cat vmlinux-arm.S | scripts/checkstack.pl arm
0x0012c858 nlmclnt_reclaim [vmlinux-arm.o]:             720
0x0025748c do_tcp_getsockopt.clone.11 [vmlinux-arm.o]:  552
0x00258d04 do_tcp_setsockopt.clone.14 [vmlinux-arm.o]:  544
0x000b2db4 do_sys_poll [vmlinux-arm.o]:                 532
0x00138744 semctl_main.clone.7 [vmlinux-arm.o]:         532
0x00138ec4 sys_semtimedop [vmlinux-arm.o]:              484
0x000c5618 default_file_splice_read [vmlinux-arm.o]:    436
0x00251de4 do_ip_setsockopt.clone.22 [vmlinux-arm.o]:   416
0x00191fd4 extract_buf [vmlinux-arm.o]:                 408
0x0019bc24 loop_get_status_old [vmlinux-arm.o]:         396
0x000e6f88 do_task_stat [vmlinux-arm.o]:                380
0x0019b8f0 loop_set_status_old [vmlinux-arm.o]:         380
0x002078f0 snd_ctl_elem_add_user [vmlinux-arm.o]:       376
0x0026267c tcp_make_synack [vmlinux-arm.o]:             372
0x00127be4 nfs_dns_parse [vmlinux-arm.o]:               368
0x000b2240 do_select [vmlinux-arm.o]:                   340
0x001f6f10 mmc_blk_issue_rw_rq [vmlinux-arm.o]:         340
0x001726a0 fb_set_var [vmlinux-arm.o]:                  336
0x000c58d0 __generic_file_splice_read [vmlinux-arm.o]:  316
0x0022a074 dev_seq_printf_stats [vmlinux-arm.o]:        316
0x0006383c tracing_splice_read_pipe [vmlinux-arm.o]:    308
0x000c53c8 vmsplice_to_pipe [vmlinux-arm.o]:            308
0x002512b4 do_ip_getsockopt [vmlinux-arm.o]:            304
0x00225f68 skb_splice_bits [vmlinux-arm.o]:             300
```

stack_size

Below are some results for static analysis of function stack depth in the Linux kernel, using 'stack_size'. (stack_size is a custom tool written by Tim Bird, before he found out about checkstack.pl.)

See this kernel message for a patch containing 'stack_size': https://lkml.org/lkml/2011/10/18/479

ARM results

The following results include the reduction in size for 'struct poll_wqueue':

```
$ ./stack_size vmlinux-arm
============ RESULTS ===============
number of functions     = 14371
max function stack depth= 736
function with max depth = nlmclnt_reclaim

Function Name                   Stack Depth
=====================           ===========
__generic_file_splice_read        352
do_select                         376
loop_set_status_old               392
snd_ctl_elem_add_user             408
extract_buf                       432
default_file_splice_read          472
sys_semtimedop                    520
semctl_main.clone.7               560
do_sys_poll                       568
nlmclnt_reclaim                   736
```

x86_64 results

```
$ ./show_stacks_x86_64.py vmlinux-x86_64.o
============ RESULTS ===============
number of functions     = 29587
max function stack depth= 1208
function with max depth = security_load_policy

Function Name                   Stack Depth
=====================           ===========
x86_schedule_events               632
drm_crtc_helper_set_mode          632
sys_semtimedop                    664
do_task_stat                      712
node_read_meminfo                 760
default_file_splice_read          792
do_select                         920
nlmclnt_reclaim                   936
do_sys_poll                      1048
security_load_policy             1208
```

CONFIG_DEBUG_STACK_USAGE

There is kernel feature to output the stack usage of each process, as well as the process that uses the most stack in the system. This is controlled by the kernel configuration option CONFIG_DEBUG_STACK_USAGE.

This option modifies the process creation path so that the stack is initialized with all zeros. At any time, a request can be made to measure the stack depth of all running processes. This is calculated by measuring the amount of zeros from the end of thread_info to the first non-zero item on each stack.

In more detail, it does the following:

- at process creation time, fills the stack with zeros (kernel/fork.c)
- on sysrq 't', show free space, from call to stack_not_used() (kernel/sched.c)
  - it shows as 0 otherwise ??
- define check_stack_usage(), which emits printks on each low-water hit
  - low-water appears to be global over all stacks
  - check_stack_usage() is only called on process exit, so you might not know about a problem process until very late
- stack_not_used() is defined in include/linux/sched.h. It counts the number of zero bytes following the end of thread_info going up.

As the systems runs, any time the stack low-water mark is exceeded, then the kernel prints a report (logs it to the kernel message log). This can be viewed with the dmesg command:

Here is example output, greping the kernel message log for "greatest":

```
# dmesg | grep greatest
kworker/u:0 used greatest stack depth: 10564 bytes left
busybox used greatest stack depth: 9512 bytes left
busybox used greatest stack depth: 9504 bytes left
grep used greatest stack depth: 9372 bytes left
init used greatest stack depth: 9028 bytes left
```

To get a report on the stack usage of currently running processes, you use 't' with sysrq. For example:

```
$ echo t >/proc/sysrq-trigger
```

A stack dump for each process is shown, along with stack usage information.

Here is some sample output:

```
$ echo t >/proc/sysrq-trigger
$ dmesg | grep -v [[]
  task                PC stack   pid father
init            S 802af8b0   932     1      0 0x00000000
kthreadd        S 802af8b0  2496     2      0 0x00000000
ksoftirqd/0     S 802af8b0  2840     3      2 0x00000000
kworker/0:0     S 802af8b0  2776     4      2 0x00000000
kworker/u:0     S 802af8b0  2548     5      2 0x00000000
migration/0     S 802af8b0  2704     6      2 0x00000000
migration/1     S 802af8b0  2704     7      2 0x00000000
kworker/1:0     S 802af8b0  2560     8      2 0x00000000
ksoftirqd/1     S 802af8b0  3024     9      2 0x00000000
khelper         S 802af8b0  2824    10      2 0x00000000
sync_supers     S 802af8b0  2872    11      2 0x00000000
bdi-default     S 802af8b0  2584    12      2 0x00000000
kblockd         S 802af8b0  2824    13      2 0x00000000
khubd           S 802af8b0  2744    14      2 0x00000000
rpciod          S 802af8b0  3024    15      2 0x00000000
kworker/0:1     S 802af8b0  1240    16      2 0x00000000
kswapd0         S 802af8b0  2848    17      2 0x00000000
fsnotify_mark   S 802af8b0  2632    18      2 0x00000000
nfsiod          S 802af8b0  3024    19      2 0x00000000
kworker/u:1     S 802af8b0  2840    20      2 0x00000000
hoge            S 802af8b0  3024    23      2 0x00000000
kworker/1:1     S 802af8b0  1716    24      2 0x00000000
flush-0:13      S 802af8b0  2528    28      2 0x00000000
telnetd         S 802af8b0  1848    48      1 0x00000000
ash             R running   1264    56      1 0x00000000
```

Stack tracer in ftrace

For detailed instructions, see: Ftrace#Find_deepest_kernel_stack

Rough notes:

- Turn on CONFIG_STACK_TRACER in kernel config
- pass 'stacktrace' on kernel command line, or at runtime do:
- echo 1 >/proc/sys/kernel/stack_tracer_enabled
- mount -t debugfs none /sys/kernel/debug
- \
- cat /sys/kernel/debug/tracing/stack_trace

See http://lwn.net/Articles/295955/

Stack limit patches

Sony has a series of patches which implement a stack guard page, and use that to show a backtrace if the process uses more than a specified amount in its kernel stack. In essence, this creates a hard failure for a controlled stack overflow event.

These patches do the following: Add the config options:

- CONFIG_SNSC_DEBUG_STACK_LIMIT - perform stack layout changes
- CONFIG_SNSC_SUPPORT_4KB_MAPPING - re-map kernel memory for 4K TLB mappings
- CONFIG_SNSC_DEBUG_STACK_LIMIT_MANUAL - allow a user-specified starting stack size

the patches do several things:

- change the stack layout to place thread_info at the top of the stack rather than at the bottom
- change the stack size to 16KB (order 2)
- allow configuring the default starting position of the stack, to simulate an arbitrary stack size (default is right below thread_info)
- remap the kernel memory so that 4KB mappings are used
  - this allows unmapping the bottom page of the stack, so that a fault occurs when the page is accessed (on a stack overflow for a manually configured small stack)
  - this is only turned on if you specify 'use_4kb_mapping' on the
- unmap the bottom page of the stack (the guard page)
  - this is only turned on if you specify 'unmap_stack'

[FIXTHIS - need to add more to this section]

ARM 4K Stacks

In October of 2011, Tim Bird submitted patches to add 4K stack support for the ARM architecture to the Linux kernel. The patches he submitted are here:

- https://lkml.org/lkml/2011/10/18/476
  - ARM 4Kstacks: introduction
- https://lkml.org/lkml/2011/10/18/477
  - ARM 4Kstacks: Add support for 4K kernel stacks to ARM
- https://lkml.org/lkml/2011/10/18/479
  - ARM: Add static kernel function stack size analyzer, for ARM
- https://lkml.org/lkml/2011/10/18/481
  - ARM 4Kstacks: Decrease poll and select stack usage, when using 4K stacks

After some discussion, these patches were not accepted into mainline.

The following points were problems that needed to be addressed for this patch set:

```
* Should make this depend on CONFIG_EXPERT (to warn developers who attempt to use this)
* Should add interrupt stacks to ARM to take pressure off of 4K stacks
* Should determine if 4K stacks use case will cause ripple effect and lots of ifdefs and hard maintenance issues thro
ughout the kernel.  In particular, need to look at:
  * %pV recursion in printk.  This is used by several file systems
    * question: for operation or just reporting??
```

Dave Chinner (here) wrote:

```
There's a good reason 4k stacks went away: it's
simply not enough space for the deep 60+ function call stacks we see
with even trivial storage stack configurations.

The stack usage on 32 bit ARM and x86 is going to be similar, so
you're going to be fighting a losing battle - there is no stack
space that can be trimmed from most paths. To make matter worse,
there's been stuff done to the storage stack that significantly
increases stack usage since 4k stacks went away (e.g. the on-stack
block plugging changes).

And FWIW, XFS is widely used on ARM based NAS devices, so this isn't
a theoretical problem I'm making up here...
```

This is a pretty good example of people denying a use case with a red herring.

Possible mixed stack size feature

One option for realizing most of the benefits of 4K stacks, while preserving more robustness, would be to utilize mixed stack sizes in the kernel.

Processes known to exercise only certain, stack-conservative, code paths in the kernel could utilize 4K stacks, and other processes could utilize 8K (or larger) stacks for safety purposes.

There would have to be a mechanism to support selecting the stack size at process creation time. One simple mechanism would be to introduce a child_stack_size parameter in thread_info, settable via /proc, and use this on the clone system call.

A system to support different-sized stacks by changing the stack size of already running processes would likely be too complicated to be practical.

Currently, the method of accessing the thread_info structure for a task in the kernel relies on the stack size of all processes being consistent among all processes (and being a power of two). A pointer to thread_info is obtained by masking the current stack pointer with a value dependent on the (global) size of the stack. With mixed stack sizes, a different mechanism would be needed to convert from stack pointer to thread_info. One method which might work would be to pre-allocate a stack pool for non-standard-sized stacks, and use pointer comparison to see if SP fell within the pool. If the pool was exhausted, the default stack size would be used.

This would work best in the case of a system with an identifiable number of processes which would use special-sized stacks.

Problems

This area has random notes for stack depth management issues:

Problem routines

do_select, do_sys_poll

The structure 'struct poll_wqueue' is a large data structure used for the select() and poll() system calls to manage a sub-set of the file descriptors being polled. This structure includes an array of wait queues which can be used immediately (without requiring or waiting for a memory allocation) for polling file I/O.

The number of entries in the array of wait queues can be controlled via macros in include/linux/poll.h

nlm_clnt_reclaim

network lock manager for network filesystems. Not applicable to most embedded products (except possibly during development).

security_load_policy

An selinux routine, not applicable to embedded.

> From: eLinux.org

Linux Tiny

Contents

Introduction

The goal of the linux-tiny project is to reduce the memory and disk footprint of the mainstream Linux kernel, as well as to add features to aid working on small systems. Target users are developers of embedded system and users of small or legacy machines such as 386s and handheld devices.

Patch releases against the mainstream Linux kernel have been discontinued. Instead of spending a valuable amount of time carrying patches forward from one kernel version to the next, we chose to focus on a few patches and spend our time trying to get them merged into the mainline kernel.

Visit the FAQ for more information.

Resources

Patches

- Available on Gitorious (HTTP, GIT). This repository is currently not maintained.

Mailing Lists

- Archive of linux-tiny

Presentations

- Kernel Size Report presentation by Matt Mackall at Embedded Linux Conference in April 2008. Video is available.
- Linux-tiny presentation by Thomas Petazzoni (Free Electrons) at Embedded Linux Conference in April 2008. Video is available.
- Linux-tiny presentation by Michael Opdenacker (Free Electrons) at Embedded Linux Conference Europe in November 2007.

- http://elinux.org/images//8/83/Pdf.gif Linux-tiny revival http://elinux.org/images/d/da/Info_circle.png presentation given by Tim Bird at the Japan Technical Jamboree #16 in August 2007
- Linux-tiny Presentation by Matt Mackall, delivered at CELF's Technical Conference in April 2005.

News

- A LWN.net article is at: LWN Article (Sep 2007), By Jake Edge
- Michael Opdenacker was announced as the new maintainer.
  - See the project revival message to the kernel mailing list
- A Linux Weekly News article (2003) about the project is at: LWN Article

Old patch releases

Old release downloads

- Linux 2.6.23.0:
  - Media:Tiny-quilt-2.6.23-0.tar.bz2
- Linux 2.6.22.5:
  - Media:Tiny-quilt-2.6.22.5-1.tar.gz
- Linux 2.6.22.1:
  - Media:Tiny-quilt-2.6.22.1-1.tar.gz
- Linux 2.6.16.19
  - broken-out patchset: 2.6.16.19-tiny1-broken-out
  - Porting notes: notes
- Linux 2.6.0 to 2.6.14:
  - Older Linux-tiny patchsets can be downloaded from: Linux Tiny Patchset

Installation Instructions

These instructions were for the Linux-tiny release for 2.6.23. Adjust accordingly for a different kernel version.

To apply the above patches, you need the referenced kernel (2.6.23) and quilt (overview). Follow these steps:

```
$ wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.23.tar.bz2
$ tar -xjf linux-2.6.23.tar.bz2
$ wget http://elinux.org/images/3/3c/Tiny-quilt-2.6.23-0.tar.bz2
$ cd linux-2.6.23
$ tar -xjf ../Tiny-quilt-2.6.23-0.tar.bz2
$ quilt push -a
$ cd ..
$ mv linux-2.6.23 linux-2.6.23-tiny1
```

Auxiliary tools (for Linux-tiny developers)

Here is a short shell script for making a tiny-quilt release: Media:release-tiny

How to use

In the Kernel Size Tuning Guide, there is a lot of information about how to measure kernel size, and how to use the kernel patches and configuration items provided by Linux-tiny.

Test Project and Results

There is a test for Linux-tiny (and kernel configuration option results, in general). Some test results from this test were previously available from the CE Linux Forum test lab, at: http://testlab.celinuxforum.org/otlwiki/ConfigSizeTestResults (link is now broken).

The CELF System Size working group has worked extensively with the Linux-tiny patch set. There is a script to produce a report of size reductions for the individual patches in the patchset, and results from various vendors about their use of Linux-tiny. See Linux Tiny Test Project

Old usage notes

There are some miscellaneous usage notes at: Linux Tiny Notes

Ideas and patch candidates

See Linux Tiny Patch Ideas

Original Announcements and e-mail

The original (Dec 11, 2003) announcement about the patchset, to the kernel mailing list, is available here:

- Announcement

Here are some other announcements from Matt Mackall to LKML:

- 2.6.0-tiny1 Dec 27, 2003
- 2.6.1-rc1-tin1 Jan 2, 2004
- 2.6.1-rc1-tiny2 Jan 6, 2004

Recent discussion thread on lkml is summarized at: LKML Thread Summary. The thread is available at here.

Note that currently, the smallest kernel that is reported in this thread is 197K compressed.

Category:

- Linux Tiny

> From: eLinux.org

Size Tunables

This page has a list of items that can be configured for the Linux kernel, which may affect the size (RAM/ROM/static/dynamic) of the kernel.

Options from Linux-tiny

CONFIG option	description	default value	value for small size	Size change (ARM)	Notes
CONFIG_CORE_SMALL	tune some kernel data sizes	N	Y	??	.
CONFIG_NET_SMALL	tune some net-related data sizes	N	Y	??	.
CONFIG_PRINTK	allow disable of printk code and message data	Y	N	??	.
CONFIG_BUG	allow elimination of BUG (and BUG_ON??) code	Y	N	??	.
CONFIG_ELF_CORE	allow disabling of ELF core dumps	Y	N	??	.
CONFIG_PROC_KCORE	allow disabling of /proc/kcore	Y	N	??	.
CONFIG_AIO	allow disabling of async IO syscalls	Y	N	??	.
CONFIG_XATTR	allow disabling of xattr syscalls	Y	N	??	.
CONFIG_FILE_LOCKING	allow disabling of file locking syscalls	Y	N	??	.
CONFIG_DIRECTIO	allow disabling of direct IO support	Y	N	??	.
CONFIG_MAX_SWAPFILES_SHIFT	number of swapfiles	5	0	??	.
CONFIG_NR_LDISCS	number of tty line disciplines	16	2?	??	.
CONFIG_MAX_USER_RT_PRIO	number of RT priority levels (schedule slots)	100	5?	??	.
Other config options	These are not in Linux-tiny, but help with size	default	small	Size change (ARM)	Notes
CONFIG_KALLSYMS	load all symbols for debugging/kksymoops	Y	N	??	.
CONFIG_SHMEM	allow use of shmem filesystem	Y	N	??	.

Options for size instrumentation

Options for measuring size	description	default value	value for instrumentation
CONFIG_KMALLOC_ACCOUNTING	turn on kmalloc accounting	N	Y
CONFIG_AUDIT_BOOTMEM	print out all bootmem allocations	N	Y
CONFIG_DEPRECATE_INLINES	cause compiler to emit info about inlines	N	Y

options that should be investigated

Option	default value	value for small size	Size change	Notes
CONFIG_SERIAL_8250_NR_UARTS	4	0	??	only useful for non-legacy ports, depends on hardware, but most embedded hardware has only legacy serial ports

```
- by default (at least on my OSK config), several different schedulers are configured
```

- only one should be used and others not compiled in.

Category:

- System Size

> From: eLinux.org

System Size Auto-Reduction

This page has notes and an outline for Tim Bird's Linux Auto-Reduction research.

Contents

Talk info

Tim gave a talk on this research at LinuxCon Japan 2013 (May 29 in Tokyo, Japan)

Title

Advanced size optimization of the Linux kernel

Abstract

This presentation will cover recent research by Tim on aggressive size reduction of the Linux kernel. This will include results from using gcc link-time optimization (LTO) with the ARM architecture (using Andi Kleen's out-of-tree patches), as well as results and discussion of other optimization techniques (including whole-system optimization for embedded devices).

This talk is directed at kernel developers interested in reducing the size of their Linux systems (and possible improving their performance in the process). The talk will be highly technical.

Final slides from talk

Media:Bird-Kernel-Size-Optimization-LCJ-2013.pdf

Research Areas

LTO

- What is it?
- what was required to get it to work?
- Andi Kleen's patch set
  - what do they do?
  - how big are they?
  - mainline status?
- what is the size gain (see ELC poster)
- what can be done with it?
- long-term possibilities for LTO

global constraints

- overall idea: create constraints external to code, and use for optimization
- rationale: can't maintain in-tree - too many config items
- make the application of constraints automatic
- use existing constraints to generate new constraints
- constraints can flow between user-space and kernel

- example: uid=0

- constraint language
- application by commenting out references (replace with 0 constant)
  - use compiler to find code references (via error messages)
    - eliminates problem with duplicate names (uid in different structure)
- constant propagation (by, e.g. LTO) reduces code

syscall elimination

- scan file system
- create report of used and unused system calls
- mark syscalls unused in kernel
  - arch/arm/kernel/calls.S (and arch/arm/kernel/entry-common.S
- make sure unused syscalls are not __attribute__(externally_visible)
  - technique of asmlinkage_\
- use LTO to eliminate calls
- results: 50K-90K

ARM stack reduction

- 4k stacks
- stack extensions

link-time rewriting

cold code compression

- D. Chanet did cold code compression
- consists of:
    - profiling the kernel
    - marking code regions as cold or frozen
    - replacing them with stubs
    - compressing them
- At execution time:
    - if a stub is called, it decompresses the code and calls it
    - stub is fixed up to directly call decompressed code in future
    - code is left decompressed forever

cold code compression

Results:

-
    - MUST see paper for details (it's quite complicated)
- on 2.4.25 kernel
    - cold code compression resulted in 7% reduction for i386 kernel and
    - 11.7% reduction for ARM kernel

Talk outline

This talk will be presented at LinuxCon Japan 2013:

Title

- Advanced size optimization of the Linux kernel
    - by Tim Bird, Sony Mobile Communication

Self-Introduction

- I am Tim Bird
- Now working at Sony Mobile
- Researching system size for many years
- Long background in extremely small systems
    - pre-professional: first program on TRS-80, in basic, 8K ram
    - NetWare Lite - file and print server in 50K (in 1991)

The problem of Bloat

- Software bloat occurs because systems are built with more software than is really needed for a given task
- Open Source software meets the needs of thousands of different systems
    - Linux scales from tiny sensors to supercomputers (extreme SMP and high-end clusters)
    - Linux supports many, many features, only some of which are configurable
- Software must be generalized for many use cases

- bloat problem is:
  - How to re-specialize the software, eliminating unused features and dead code?

Bloat (cont.)

- Software gets more generalized over time
- Can't use strategy of manual tuning (config options)
  - It gets harder and harder to remove things over time
  - About 13,000 config items now (2.6.12 had 4700)
  - You have to be an expert in too many things to reduce the kernel
- Must rely on automated methods of reduction
- Should use an additive, rather than subtractive method of building a system
  - ultimate vision: indicate what you want/need, and build up system to support it

Bloat (cont. 2)

- In desktop or server, virtual memory makes bloat issue less important for user-space programs
  - Only working set of program is loaded - pages are loaded on demand
  - For kernel, all pages are always loaded

Automatic reduction (intro)

The problem with automatic reduction is that "the system" doesn't know what software is needed and what is not. there needs to be a way to tell it about things that are not going to be used.

auto-reduce - story of 8 bytes of bloat

Story of the conditional check in kdb:

- I found a bug in kdb, when a particular option was using in the configuration file
- not everyone uses the configuration file
- not everyone uses the particular option
- bug only triggered in those circumstances
- I wrote a small patch, to guard against use of a variable prematurely
- problem: all users of KDB now have this check, and suffer this overhead
  - it wasn't much, just a single compare
  - but this is how bloat builds up over time
  - It bothered me because I knew most people didn't need the check
- "correct" solution would be to parse the config file, and make the code compile-time configurable
  - this adds more complexity than it is worth.

generalizing the problem of bloat

System doesn't know inputs:

- It's very easy to configure the kernel to omit the driver for missing hardware.
- It's very difficult to configure the kernel to omit error handling for bugs that

will never occur due to fixed use cases.

An example of fixed input (uid in kernel)

- throughout the kernel, there are references to uid
  - comparisons, storing, referencing
- it turns out this is set by setuid(), by the 'login' program.

  - login does a lookup and validates user account name in /etc/passwd
- what if /etc/passwd only has 'root' and no others?
- setuid() could only be called with a value of 0
- can I encode this constraint on the system.

types of constraints

There are numerous other examples of constraints:

- kernel command line arguments never used
- syscalls never called by any program
- parameters that are never used, or parameter values that are never passed in
  - e.g. ioctl value that is not possible
  - this only works in a fixed
- /proc values never referenced
- /sys values never referenced

also goes back up to user-space

- return values that are not possible

kernel command line args

- Documented in Documentation/kernel-parameters.txt
- defined with __setup() and early_param from include/linux/init.h
  - approximately 480 __setup routines in kernel
  - about 200 __setup_* in System.map on ARM kernel build (98 __setup_str_*)
  - about 230 early_param routines in kernel
- points to function
- almost always sets a variable, which would default to 0

- on ARM, with only console_setup and early_mem marked as 'used', there was a 19K difference in size:

(non-LTO kernel)

```
vmlinux.baseline-setup-used  =>  vmlinux-param-used
       baseline   other   change  percent
 text:  7680084  7663472   -16612   0%
 data:   362868   360516    -2352   0%
  bss:   745312   745184     -128   0%
total:  8788264  8769172   -19092   0%
```

- on ARM, with only console_setup and early_mem marked as 'used', there was a 19K difference in size:

(LTO kernel)

```
vmlinux.lto-param  =>  vmlinux.param-used
       baseline   other   change  percent
 text:  1653672  1648920    -4752   0%
 data:   131636   130244    -1392  -1%
  bss:    50688    50528     -160   0%
total:  1835996  1829692    -6304   0%
```

System.map from kernel with console_setup and early_mem as only routines marked 'used':

```
$ grep __setup System.map
c00ea4bc T __setup_irq.153323
c00f1adc t __setup_per_zone_wmarks.172539.15755
c019d570 t __setup_str_early_mem.21664.160821
c019d884 t __setup_str_console_setup.61958.160201
c019ef00 t __setup_early_mem.21659.160819
c019ef00 T __setup_start
c019ef0c t __setup_console_setup.61953.160195
c019ef18 T __setup_end
```

/proc values

- Includes sysctl values
- there are approximately 1200, NOT related specifically to a process
- about 120 per process
  - about 80 related to networking (on my desktop box)
- 40 others

Tiny Distribution

- poky-tiny distribution (yocto project)
- see https://wiki.yoctoproject.org/wiki/Poky-Tiny
- Good for testing and further research

References

- Chanet D. ... "Automated reduction of the memory footprint of the linux kernel"
- Haifen He. ..."Code Compaction of an Operating System Kernel"

Related Projects

- AnyKernel and Rumpkernel - see thesis by Antti Kantee - pooka (at) iki (dot) fi
  - https://github.com/rumpkernel/wiki/wiki
  - provides a system based on NetBSD to isolate sub-systems and drivers and allow their use in micro-kernels and user-space
  - haven't read enough of it to determine if it could be applied to Linux, but sounds like just API wrapping
  - I'm not sure how robust it would be in the context of rapid mainline churn

Materials

- File:0001-ARM-LTO-avoid-errors-on-unified-assembly-macros.patch
- File:0001-Support-elimination-of-unused-kernel-parameters.patch
- File:0001-Add-option-to-omit-unused-syscalls.patch
- Tools and notes from the project are at: https://github.com/tbird20d/auto-reduce

From: eLinux.org

Szwg Linux 26Data

Measurement Result

Renesas/SH4										
Name of Method	ROM (kernel)		RAM(kernel)		ROM (rootFS)		RAM (root FS)		Bootup Time	
	Size(KB)	Ratio	Size(KB)	Ratio	Size(KB)	Ratio	Size(KB)	Ratio	actual	Ratio
Typical Boot	654	100.0	1425	100.0	2644	100.0	0	100.0	3995	100.0
Kernel XIP	1317	201.4	245	17.2	2644	100.0	0		2082	52.1
initrd	663	101.4	1446	101.5	1276	48.3	2644		4643	116.2
cramfs	645	98.6	1443	101.3	1507	57.0	0		NA	
jffs2	690	105.5	1496	105.0	1644	62.2	0		6069	151.9
Typical Boot(2.6)	864	132.1	1679	117.8	2644	100.0	0		NA	
Kernel XIP(2.6)				0.0						
initrd(2.6)	848	129.7	1453	102.0	2644	100.0	0		NA	
cramfs(2.6)	852	130.3	1457	102.2	2644	100.0	0		NA	
jffs2(2.6)	897	137.2	1544	108.4	2644	100.0	0		NA	

NEC/VR5500A SOC										
Name of Method	ROM (kernel)		RAM(kernel)		ROM (rootFS)		RAM (root FS)		Bootup Time	
	Size(KB)	Ratio	Size(KB)	Ratio	Size(KB)	Ratio	Size(KB)	Ratio	actual	Ratio
Typical Boot	807	100.0	1637	100.0	3548	100.0	0	100.0	3494	100.0
Kernel XIP	1438	178.2	271	16.6	3548	100.0	0		2470	70.7
CompressFS(initrd)	816	101.1	1654	101.0	1249	35.2	3548		7381	211.2
cramfs	799	99.0	1653	101.0	1536	43.3	0		3494	100.0
jffs2	844	104.6	1718	104.9	1726	48.6	0		5824	166.7
Typical Boot(2.6)	897	111.2	1819	111.1	3548	100.0			2527	72.3
Kernel XIP(2.6)	1797	125.0	161	59.4	3548	100.0			1375	55.7
initrd(2.6)	901	110.4	1834	110.9	1249	100.0	3548	100.0	5484	74.3
cramfs(2.6)	888	111.1	1801	109.0	1536	100.0			2516	72.0
jffs2(2.6)	933	110.5	1899	110.5	1726	100.0			4883	83.8

Item_Table1109.xls

Platform

Editors Note: Left out original System Size Working Group content as it seems more appropriate on the CELF site. Will confirm this in near future.

Categories:

- Editor Follow Ups
- Consumer Electronics Linux
- System Size

> From: eLinux.org

File Systems

Contents

Introduction

Most embedded devices use flash memory as storage media. Also, size and bootup time are very important in many consumer electronics products. Therefore, special file systems are often used with different features, such as enhanced compression, or the ability to execute files directly from flash.

MTD

Note that flash memory may be managed by the Memory Technology Devices (MTD) system of Linux. See the MTD/Flash FAQ for more information. Most of the filesystems mentioned here are built on top of the MTD system.

UBI

The Unsorted Block Images (UBI) system in the Linux kernel manages multiple logical volumes on a single flash device. It provides a mapping from logical blocks to physical erase blocks, via the MTD layer. UBI provides a flexible partitioning concept which allows for wear-leveling across the whole flash device.

See the UBI page or UBI FAX and Howto for more information.

Partitioning

The kernel requires at least one "root" file system, onto which other file systems can be mounted. In non-embedded systems, often only a single file system is used. However, in order to optimize limited resources (flash, RAM, processor speed, boot up time), many embedded systems break the file system into separate parts, and put each part on its own partition (often in different kinds of storage.

For example, a developer may wish to take all the read-only files of the system, and put them into a compressed, read-only file system in flash. This will consume the least amount of space on flash, at the cost of some read-time performance (for decompression).

Another configuration might have executable files stored uncompressed on flash, so that they can be executed-in-place, which saves RAM and boot-up time (with a potential small loss of performance).

For writable data, if the data does not need to be persistent, sometimes a ramdisk is used. Depending on the performance needs and the RAM limits, the file data may be compressed or not.

There is no single standard for interleaving the read-only and read-write portions of the file system. This depends heavily on the set of embedded applications used for the project.

eMMC and UFS

As flash memories have gotten larger, a variety of factors has caused a shift from use of raw NAND to packaged, block-addressable NAND flash memory for embedded devices. These are chips which contain firmware on board to accept block I/O requests, similar to rotating storage media (old hard disk drives), and fullfill them. This involves mapping the read and write requests to areas of the NAND flash in the chip, and managing the NAND flash to try to optimize for correctness and longevity of the flash memory. NAND flash must be re-written in large blocks (erase blocks) that are many times the size of individual file system blocks. Therefore, the method of mapping, re-arranging and garbage collecting the allocation of blocks in the system is quite important.

These chips are run with a block-based, rather than flash-based filesystem (e.g. ext4). As of 2012, optimizing the ext4 file system for use with these systems is a hot topic area of file system research. See http://lwn.net/Articles/502472

Embedded Filesystems

Here are some filesystems designed for and/or commonly used in embedded devices, sorted in alphabetical order:

AXFS

- AXFS - Advanced XIP File System
  - Website: http://axfs.sourceforge.net/
  - This file system is designed specifically to support Execute-in-place operations. It uses a bi-phased approach. The first phase is to have the filesystem in flash and run it to collect profile data, stating what pages are used. In the second phase you build a filesystem using these profile data. This filesystem makes all pages metioned in the profile file as XIP data, which can then will be loaded to RAM upon mounting (and executed as XIP). It is also possible to put the XIP pages in NOR flash and run them from there.

Btrfs

- btrfs is a new copy-on-write filesystem that first appeared in the kernel in 2.6.29-rc1 and was merged in 2.6.30.
- Btrfs is not yet supported by many popular Linux filesystem tools such as gparted as of April 2011.
- Btrfs has been adopted as the MeeGo platform's filesystem.
- Nice Introduction Video on btrfs by Chris Mason

CramFS

- CRAMFS - A compressed read-only file system for Linux. The maximum size of CRAMFS is 256MB.
  - "Linear Cramfs" is the name of a special feature to use uncompressed file, in a linear block layout with the Cramfs file system. This is useful for storing files which can be executed in-place. For more information on Linear Cramfs, see Application XIP

F2FS

- F2FS(wikipedia entry) is a flash-friendly file system for Linux, developed by Samsung.

InitRAMFS

From March 2006 Linux Devices:

INTRODUCING INITRAMFS, A NEW MODEL FOR INITIAL RAM DISKS This clear, technical article introduces initramfs, a Linux 2.6 feature that enables an initial root filesystem and init program to reside in the kernel's memory cache, rather than on a ramdisk (as with initrd filesystems). Compared to initrd, intramfs can increase boot-time flexibility, memory efficiency, and simplicity, the author says. One especially interesting feature for embedded Linux developers is that relatively simple, deeply embedded systems can use initramfs as their sole filesystem.

http://www.linuxfordevices.com/c/a/Linux-For-Devices-Articles/Introducing-initramfs-a-new-model-for-initial-RAM-disks/

Here is a good article about how to build an initramfs:

- http://www.landley.net/writing/rootfs-howto.html

For more information, look in: Documentation/early-userspace/README

JFFS2

- JFFS2 - The Journalling Flash File System, version 2. This is the most commonly used flash filesystem.

  - The maximum size of JFFS2 is 128MB.
  - http://sourceforge.net/projects/mtd-mods has some patches by Alexey Korolev for improvements to JFFS2
    - See the presentation on Alexey's patches at:
  - To improve mount time substantially verify that the erase block summary patch is in your image. This patch is part of the jffs2 driver since 2005-09-07. A patch for an earlier version can be found at: http://www.inf.u-szeged.hu/jffs2/jffs2-summary-20050211.patch (or try your luck at http://web.archive.org/web/*/http://www.inf.u-szeged.hu/jffs2/mount.php).
  - JFFS2 has undergone improvement since early versions (~2.4.30). Modern versions of the driver in newer kernels have show stopping bugs fixed.

LogFS

LogFS was a scalable flash filesystme aimed at replacing JFFS2 for most uses.

Unfortunately, it seems to be abandoned at present.

See LogFS for details.

NFS

Due to space constraints on embedded devices, it is common during development to use a network file system for the root filesystem for the target. This allows the target to have a very large area where full-size binaries and lots of development tools can be placed during development. One drawback to this approach is that the system will need to be re-configured with local file systems (and most likely re-tested) for final product shipment, at some time during the development cycle.

An NFS client can be built into the Linux kernel, and the kernel can be configured to use NFS as the root filesystem. This requires support for networking, and mechanisms for specifying the IP address for the target, and the path to the filesystem on the NFS host. Also, the host must be configured to run an NFS server. Often, the host also provides the required address and path information to the target board by running a DHCP server.

See the the file Documentation/nfsroot.txt in the Linux kernel source for more information about mounting an NFS root filesystem with the kernel.

PRAMFS

- PRAMFS - Persistent and protected RAM File System

The Persistent/Protected RAM Special Filesystem (PRAMFS) is a full-featured read/write filesystem that has been designed to work with fast I/O memory, and if the memory is non-volatile, the filesystem will be persistent. In addition, it has Execute-in-place support.

Info on the PRAMFS specification can be found at Pram Fs Specification

Romfs

- RomFs - A small space-efficient read-only filesystem. A description can be found in Documentation/filesystems/romfs.txt or http://lxr.linux.no/linux/Documentation/filesystems/romfs.txt

SquashFS

Squash Fs is a (more) compressed read-only file system for Linux. This file system has better compression than JFFS2 or CRAMFS. After spending a long time outside of the mainline kernel, Squashfs have finally been merged and released with kernel 2.6.29.

It is possible to tune the amount of compression when running mksquashfs. The -b option allows you to specify the block size. A smaller block size generally gives less compression and a larger -b option gives more compression. However there is a downside to this. Data is read from the flash using blocks. So if you use a block size of 128k, and you need a page of 4k, still the compressed equivalent of 128k data will be read from flash. As 128k comprises 32 pages, it will result in 32 pages being read into the buffer cache, even though at the moment of reading you only need one. Often the other 31 pages will be needed as well, but if not you wasted some tiem to read and decompress the unused data. Also you got some unneeded data in the buffer cache (possibly the system even had to kick used pages from the cache in order to make room for these 31 pages).

If you care for the smallest filesystem you probably want to go with the largest block size. However, if your primary concern is performance you might want to experiment a little bit to see what works out best for you (and that could even be applying no compression at all! Mksquashfs has options: -noInodeCompression, -noDataCompression and –noFragmentCompression to control this). If you also applied function reordering (see Boot Time#User-space and application speedups a large block size will probably work out well for you.

The table below gives an idea of the amount of compression that is achieved by the various block sizes. Input was a root filesystem of an embedded device.

	size	compression
Initial	53128K	100 %
4K	17643K	33.2 %
8K	16572K	31.2 %
16K	15780K	29.7 %
32K	15204K	28.6 %
64K	14812K	27.9 %

A presentation on Squash FS by Phillip Lougher at ELC Europe 2008: slides and video.

UBIFS

UBIFS is a flash-based filesystem, implemented on top of the Unsorted Block Images (UBI) interface.

It has good performance compared to Jffs2 and yaffs.

Please see the UBIFS page for more details.

YAFFS2

- YAFFS - Yet Another Flash File System - a file system designed specifically for NAND flash.

YAFFS2 is simple, portable, reliable and self-contained. It is widely used in embedded OSes other than Linux, and can also be used stand-alone without an OS, e.g. in bootloaders. When used with Linux it can use MTD or its own flash driver. Similarly it can use the VFS or its own posix layer. It is log-structured, and single-threaded. It does not do compression itself - either compress the data itself or use squashfs on top of YAFFS2.

YAFFS2 is designed to boot quickly (insofar as a log-structured FS that has to scan the flash can). It uses checkpointing so that if a partition was unmounted cleanly then there is no need to rescan the flash on power-up. All the features of the FS are configuable so you can trade off things like maximum file/partition size, flash block size, file granulaity etc. Data is

written straight through to the flash except for caching to ensure efficienct use of blocks. YAFFS2 normally uses the OOB are of the flash for its metadata, allowing faster booting as only the OOB needs to be read for flash scan. It can keep its metadata inside the main page area at the expense of some speed.

Despite having been in use on Linux in real products since 2004 it has not yet made it to the mainline.

-
    - Presentation on YAFFS2 by Wookey at ELC Europe 2007: yaffs.pdf
    - Presentation from CELF Jamboree 17 comparing YAFFS and JFFS2 on 2.6.10: celf_flash.pdf

YAFFS2 is GPLed, but is also available under dual-licensing terms for use in non-free contexts from Aleph One Ltd.

Mounting the root filesystem

The root filesystem is mounted by the kernel, using a kernel command line option. Other file systems are mounted from user space, usually by init scripts or an init program, using the 'mount' command.

The following are examples of command lines used for mounting a root filesystem with Linux:

- Use the first partition on the first IDE hard drive:
    - root=/dev/hda1
- or in later kernels:
    - root=/dev/sda1
- Use NFS root filesystem (kernel config must support this)
    - root=/dev/nfs

(Usually you need to add some other arguments to make sure the kernel IP address gets configured, or to specify the host NFS path.)

- Use flash device partition 2:
    - root=/dev/mtdblock2

```
[FIXTHIS - should probably mention initrd's here somewhere]
```

Mounting JFFS2 image on PC using mtdram

Since it is not possible to use the loopback device to mount JFFS2 images, mtdram needs to be used instead. Usually three modules are needed to get it working:

- mtdram: Provides an MTD partition in RAM. The size can be defined with the total_size parameter in kilobytes.
- mtdblock: This will create a block device for access to the partition.
- jffs2: Since JFFS2 is usually not used as a filesystem on a PC, support needs to be loaded manually.

```
modprobe mtdram total_size=16384
modprobe mtdblock
modprobe jffs2
```

Depending on the target's endianess the image file might need conversion to PC endianess. jffs2dump from the MTD tools can be used to archive this.

```
jffs2dump -b -c -e <output-filename> <input-filename>
```

The final image can be copied to the block device using dd.

```
dd if=<image-file> of=/dev/mtdblock0
```

Mounting is done in the usuall way.

```
mount /dev/mtdblock0 /tmp/jffs2 -t jffs2
```

Mounting UBI Image on PC using nandsim

First create a simulated NAND device (this one is 256MB, 2048 page size). _id_byte= corresponds to the ID bytes sent back from the NAND.

```
$ sudo modprobe nandsim first_id_byte=0x20 second_id_byte=0xaa third_id_byte=0x00 fourth_id_byte=0x15
```

Check it was created.

```
$ cat /proc/mtd
dev:    size   erasesize  name
mtd0: 10000000 00020000 "NAND simulator partition 0"
```

Next, attach it to a mtd device.

```
$ sudo modprobe ubi mtd=0
```

I had to detach it prior to formatting it.

```
$ sudo ubidetach /dev/ubi_ctrl -m 0
```

If that ubidetach step fails when you enter it, just proceed to the next step to format the mtd device.

```
$ sudo ubiformat /dev/mtd0 -f <image>.ubi
ubiformat: mtd0 (nand), size 268435456 bytes (256.0 MiB), 2048 eraseblocks of 131072 bytes (128.0 KiB), min. I/O size
 2048 bytes
libscan: scanning eraseblock 2047 -- 100 % complete
ubiformat: 2048 eraseblocks have valid erase counter, mean value is 1
ubiformat: flashing eraseblock 455 -- 100 % complete
ubiformat: formatting eraseblock 2047 -- 100 % complete
```

Then, attach it.

```
$ sudo ubiattach /dev/ubi_ctrl -m 0
UBI device number 0, total 2048 LEBs (264241152 bytes, 252.0 MiB), available 0 LEBs (0 bytes), LEB size 129024 bytes
(126.0 KiB)
```

Make a target directory, and mount the device.

```
$ mkdir temp
$ sudo mount -t ubifs ubi0 temp
```

Issues with General Purpose filesystems used in embedded

MMC/sdcard card characteristics

MMCs and SDcards are flash devices which present a block-oriented interface to their host computer. Often, these devices are used in embedded devices and have characteristics that are tuned for block access using a FAT filesystem. But they are presented at "black boxes", with internal logic and algorithms that are not exposed to the host computer.

Some work is in progress to survey characterize these attributes, and to adapt Linux to be able to use these devices more efficiently.

See https://wiki.linaro.org/WorkingGroups/KernelConsolidation/Projects/FlashCardSurvey

and https://wiki.linaro.org/WorkingGroups/KernelConsolidation/Projects/FlashDeviceMapper (These projects appear to be the work of Arnd Bergmann)

Special-purpose Filesystems

ABISS

The Active Block I/O Scheduling System is a file system designed to be able to provide real-time features for file system I/O activities.

See ABISS

Layered Filesystems

Layered filesystems enable you to mount read-only media and still have the possibility to write to it. At least, the writing part will end up somewhere else, which is transparantly handled by the layered filesystem. It has been around for quite some time and below are some examples of filesystems already usable on (embedded) Linux systems out-of-the-box.

UnionFS

Sometimes it is handy to be able to overlay file systems on top of each other. For example, it can be useful in embedded products to use a compressed read-only file system, mounted "underneath" a read/write file system. This give the appearance of a full read-write file system, while still retaining the space savings of the compressed file system, for those files that won't change during the life of the product.

UnionFS is a project to provide such a system (providing a "union" of multiple file systems).

See http://www.filesystems.org/project-unionfs.html

See also union mounts, which are described at http://lkml.org/lkml/2007/6/20/18 (and also in Documentation/union-mounts.txt in the kernel source tree - or will be, when this feature is merged.)

aufs

Another UnionFS. Go to http://aufs.sourceforge.net for more details.

mini_fo

minifo = mini fanout overlay file system.

Go to http://www.denx.de/wiki/Know.MiniFOHome for more details.

Apparently this is not maintained any more. Last information is from 2005.

Performance and benchmarks

Tools to measure performance

For a very simple disk performance measurement, you can use 'dd'. The following writes a 2G file of all zeros to a filesystem, then clears the page cache, and reads the file back:

- dd if=/dev/zero of=test bs=1048576 count=2048
- sync
- sudo echo 3 >/proc/sys/vm/drop_caches
- dd if=test of=/dev/null bs=1048576

You can also use IOZone to measure the performance of a Linux filesystem.

See http://www.iozone.org/

Some benchmark systems that are commonly used with desktop linux are

- bonnie
- dbench
- Portable, fully-threaded I/O benchmark program (tiobench)
- Flexible File System Benchmark (ffsb)

Comparison of flash filesystems

Cogent Embedded tests (2013)

This section has links to benchmarks, testing and tuning information.

- eMMC/SSD Filesystem Tuning Methodology v1.0 document
  - Contains testing methodology, and results (performance and robustness) for tuning different filesystems (btrfs, ext3, and f2fs) on different flash media

Free Electrons tests (2011)

In 2011, the CE Linux Forum contracted with Free Electrons to perform systematic testing of multiple flash filesystems over multiple kernel versions.

The results are here: Flash_Filesystem_Benchmarks

Other projects

Multi-media file systems

- XPRESS file system - [See OLS 2006 proceedings, presentation by Joo-Young Hwang]
  - I found out at ELC 2007 that this FS project was recently suspended internally at Samsung

WikipediaFS

A mountable virtual filesystem that allows accessing mediawiki based sites as regular files using a regular editor. Currently this filesystem is unmaintained. See http://wikipediafs.sourceforge.net/ for more info.

wikifs

This one seems similar to WikipediaFS, but aimed at Plan9 and inferno. See http://www.cs.bell-labs.com/magic/man2html/4/wikifs for more info.

Category:

- File Systems

> From: eLinux.org

AXFS

Contents

Description

The Advanced XIP File System is a Linux kernel filesystem driver that enables files to be executed directly from flash or ROM memory rather than being copied into RAM.

This project was started by engineers working at Intel. Since that the key developers have moved over to Numonyx. AXFS is intended to be a replacement for Linear XIP CRAMFS, and combines features from CRAMFS and SquashFS. The advantage over existing solutions is that it can provide reduced RAM with less flash because it is possible to compress only those pages that contribute to RAM savings. It includes tools to identify pages that should be uncompressed. It is Beta quality as of July 2006.

Rationale

This feature is important because XIP is a very common method of conserving RAM on an embedded system.

Furthermore some boot time reduction can be achieved.

Resources

Project site and contacts

The main project site is at: axfs

Those interested can send an email using the interface at axfs contact and they will be informed when the final version is released.

Documents

- Poster (in powerpoint) describing AXFS
- ELC 2006 Presentation on XIP by Jared Hulbert
- Demystifying Embedded Code Storage: Optimizing for Lower Cost and Higher Performance through Balanced XIP - White paper by Intel comparing different XIP approaches
- Numonyx™ AXFS Technical Documents
  - contains the users guide and the image builder guide
- Talk *Advanced XIP filesystem* by Jared Hulbert at the Ottawa Linux Symposium 2008. Paper and video

Specifications

See Kernel XIP Specification R2

Downloads

Patch

See http://axfs.sourceforge.net/.

Utility programs

There are two utility programs:

- mkfs.axfs described here is a simple filesystem builder. It compresses using 4K blocks and accepts profile data in .csv format (and perhaps also in xml format). This differs from what is described in the document. The program generates a single image file with both the compressed and uncompressed data. Upon mounting the image the uncompressed XIP data are copied to RAM. This program is GPL.
- A Numonyx proprietary image builder described here. Judging from the documentation (-c option) this builder provides larger pages (and hence better compression) similar to the larger pages that are used by SquashFS. Apparently it can also generate two output files (-b and -o options). I assume that way you get two images. One with the compressed data, one with the XIP data. That would allow e.g. putting the compressed data in NAND flash and the XIP data in NOR flash or RAM.

How To Use

- Download the patches from the svn archive as described on this project page.
- Apply the patches as described here.
- In the config file two new configuration options will show up. One is called CONFIG_AXFS and includes the AXFS filesytem. The second one is called CONFIG_AXFS_PROFILING and enables the profiling. You need to enable both at first.
- Make an axfs filesystem (using mkfs.axfs) and write that image to flash. (e.g. using nandwrite).
- Boot your new kernel.
- Run whatever applications (from the AFXS partition) you want to have in XIP. If you are interested in boot improvement these are typically the things run from init.
- Internally AXFS records the page accesses. Once you ran all programs you want to have in XIP, you can copy /proc/axfs/volumeX to a file (where X is typically 0).
- If desired you can remove things from the file. Note that the file consists of lines with a filename, an offset and a counter. This describes the page that is put in XIP. You'll notice that not all of the file is put in XIP. Only those pages that are actually called are put in the XIP section.
- Use the file from the previous version as input for mkfs.axfs. This will generate a file with both XIP and non-XIP data

(the former uncompressed, the latter compressed).

- Write this file to flash, and use it. You'll see an increase in mount time, as the XIP data is copied to RAM at mount time. However the executables should start faster as no data need to be read from flash or copied in RAM. B.t.w. it is possible to have your root filesystem as axfs filesytem by using the kernel boot parameter "rootfstype=axfs".
- To reduce size and increase performance rebuild the kernel with profiling (CONFIG_AXFS_PROFILING) turned off.

How to validate

[put references to test plans, scripts, methods, etc. here]

Sample Results

No sample results are available yet.

However from studying the documentation and knowing how XIP works, a boot peformance gain can be expected when running from NOR flash. The amount of gain cannot really be qualified because it depends on how often the code is executed. E.g. consider an often executed loop in NOR. The advantage with NOR flash and XIP is that you do not need to load the page (so you save some time). However as NOR is slower there will be a small penalty during every instruction executed from NOR. So if you have a loop that is executed very often the cumulative penalty will exceed the gain from not having to load the page. Of course if you know that this is the case you can exclude this page from XIP.

mkfs.axfs does not support generating two separate images. Therefore in the open source scenario the only way to test this is by putting both the XIP and non-XIP data in NOR.

Alternatively it is possible to put the generated image in NAND. In that case the XIP data is loaded during mounting. Effectively this does not gain much. You only shift the reading from program execution to mounting. Of course there can be a small saving since no in-memory copying is needed. Benchmarks need to be performed to actually quantify the gain.

With the proprietary image builder it seems possible to create two images, one with XIP and one without XIP. This allows for situations where the XIP data is in NOR and the non-XIP data is in NAND. Alternately the XIP data could be in RAM and loaded by the boot loader. It might be possible that this gives some gain (as the boot loader might have less overhead reading a file than linux has). Again this need to be benchmarked.

Case Study 1

Case Study 2

Status

Jared Hulbert of Intel writes (in July 2006):

The filesystem is working well. We optimized an Opie Linux build using cramfs and axfs. The resulting images where 49MB XIP cramfs, XIP axfs used only 39MB, and 34MB for a fully compressed cramfs. We're polishing it up for release hopefully in the next couple weeks. We are also debugging a 2.4 port which we will make available as well.

In august 2008, the project submitted to LKML for the second time. On mid nov 2008, the project svn archive contained patches for 2.4.21 and 2.6.10 to 2.6.27.

- Status: [not started??]
  - (one of: not started, researched, implemented, measured, documented, accepted)
- Architecture Support:
  - (for each arch, one of: unknown, patches apply, compiles, runs, works, accepted)
    - i386: unknown

    - ARM: unknown
    - PPC: unknown
    - MIPS: mostly works (some open issues and more testing needed)
    - SH: unknown

Future Work/Action Items

Here is a list of things that could be worked on for this feature:

- More testing/maturing
- An improved open source mkfs.axfs that provides separate images and compression using larger blocks than 4K.
- Benchmarking

Category:

- File Systems

> From: eLinux.org

F2FS

F2FS is a flash-friendly filesystem, developed by Samsung, and introduced into version 3.8 of the Linux kernel.

Here is some information on it:

- wikipedia entry for F2FS
- Presentation given at ELC, in February, 2013
    - Flash-Friendly File System (PDF)
- Presentation given at ELC Europe, in November 2012
    - A New File System Designed for Flash Storage in Mobile (PDF)
- Samsung's F2FS Filesystem - LWN.net article from October of 2012
- An f2fs teardown - technical analysis by Neil Brown on LWN.net
    - This article has very good technical details explaining the pros and cons of the filesystem.
    - Also, some of the comments to the article have good information.

> From: eLinux.org

Flash Filesystem Benchmarks

Free Electrons has performed flash filesystem benchmarks, with funding from the CE Linux Forum. This page is the starting page to present the methodology and the results of these benchmarks.

Contents

Test methodology

Free Electrons created Python scripts that automate the execution of commands through a serial line (including bootloader and kernel booting), and measure the time taken to execute these commands. The scripts were designed to be generic, and support for new boards can easily be added by creating board specific Python definitions. The complete details of what tests are performed and how measurements are made are available in the Flash Filesystem Benchmarks Protocol page.

The current version of these scripts can be found in a Git repository on gitorious, and are released under the terms of the GPLv2 license. Working board automation files are provided for the CALAO USB-A9263-C02 and IGEPv2 boards. You will need to build a root filesystem to run the tests on and create filesystems of different sizes (8, 32, 252 and 508 MB) to be tested ; both of which that have been used by Free Electrons will be available soon.

Results

Comparison of different versions of the Kernel

See Flash Filesystem Benchmarks Kernel Evolution to find possible regressions

Linux 3.1 results

See Flash Filesystem Benchmarks 3.1 for the results

Linux 3.0 results

See Flash Filesystem Benchmarks 3.0 for the results

Linux 2.6.39 results

See Flash Filesystem Benchmarks 2.6.39 for the results

Linux 2.6.38 results

See Flash Filesystem Benchmarks 2.6.38 for the results

Linux 2.6.36 results

See Flash Filesystem Benchmarks 2.6.36 for the results

Presentations of the results

Previous results of those benchmarks were presented:

- At ELC Europe 2010, the slides (PDF) are available
- At ELC Europe 2008, the slides (PDF) are available

Details on the hardware platforms used

- CALAO USB-A9263-C02
  - AT91SAM9263 processor at 200 Mhz
  - 64 MB of RAM
  - 256 MB of NAND Flash from Samsung K9F2G08U0A
- IGEPv2 boards
  - DM3730 processor at 1 Ghz
  - 512 MB of RAM
  - 512 MB of dual-plane SLC OneNAND Flash from Numonyx NAND04GR4E1A

Category:

- Flash Filesystem Benchmarks

> From: eLinux.org

Linux Devices

a web site targeting information for running linux on embedded devices

http://www.linuxdevices.com/

Category:

- Important Websites

> From: eLinux.org

LogFS

LogFS is a raw flash filesystem, intended to replace JFFS2, with a focus on scalability.

In July of 2007, Matt Mackall wrote:

```
LogFS is a filesystem designed to support large volumes on FLASH. It
uses a simple copy-on-write update process to ensure consistency (the
"log" in the name is a historical artifact). It's easily the most
modern and scalable open-source FLASH filesystem available for Linux
and it's well on its way to being accepted in the mainline tree.
```

It was originally written by Joern Engel, but has recently (up to October, 2012) been maintained by Prasad Joshi.

LogFS was mainlined in kernel version 2.6.34 (in November 2009).

It appears to be less-used than other flash filesystems, and in some testing was not robust enough to complete testing.

Resources

- Home page: http://logfs.org/logfs/
  - Mailing list: http://logfs.org/cgi-bin/mailman/listinfo/logfs
  - NOTE: this site appears down as of October 2012
- Articles:
  - LogFS article, LWN.net, May 2007
  - LogFS: A new way of thinking about flash filesystems Linux.com, May 2007

Scott Preece wrote:

```
The big win for LogFS (in my limited knowledge of it) is that it stores
its tree structure in the media, rather than building it in memory at
mount time. This significantly reduces both startup time and memory
consumption. This becomes more important as the size of the flash device
increases.
```

Some newer flash memory, like MLC (multi-level cell), are not well supported.

> From: eLinux.org

Pram Fs

Contents

Introduction

This page describes the Protected RAM File System (PRAM FS) feature.

PRAM FS is a file system that enhances the security of system data in the presence of kernel bugs or rogue programs.

The protected RAM file system will ordinarily remain consistent even if kernel data pointers are corrupted, or if the kernel starts executing unexpectedly in the wrong location. This is accomplished by making the RAM pages used by PRAM FS non-writable except during the actual file operations themselves.

Rationale

A single bug in the Linux kernel may cause catastrophic damage to a system. If a product holds irreproducible security keys, financial data, or account information, then loss of such data could render the product unusable, or worse. The customer could suffer financial or legal harm (from account theft or identity theft).

It is not possible to guarantee with certainty that there are no bugs in the Linux kernel. However, it is possible to decrease the probability that a bug in the kernel will cause damage to a particular area of memory or storage. This protected area can then be used, with greater confidence, to hold sensitive user or product data.

References

The home page for the PRAMFS project is at: http://pramfs.sourceforge.net/

That site contains a LOT of detailed technical information and more explanation of the rationale for this feature.

Downloads

Patch

- See celinux-dev archive message 197 for a submission to CELF in 2004)
- Patches for 2.6.30 were posted to lkml in June 2009 - see http://lkml.org/lkml/2009/6/13/86

Utility programs

Pram fs can be created and populated using normal Linux filesystem utilities.

How To Use

See the file `Documentation/filesystems/pramfs.txt` for instructions on its use (once the patch is applied).

Status

Pramfs was submitted for consideration for inclusion in the 2.6.4 kernel, in March 2004. There was a thread of discussion here

There were a few, easily answered, concerns raised. But the patch was not accepted into mainstream.

I talked to Andrew Morton about this in April, 2004, and he said the threshold is high for getting a new filesystem into the mainline kernel, because each filesystem adds incremental, ongoing, source maintenance overhead.

Sample Results

Here there are some benchmark results made with bonnie++. The board used was an Atmel ngw100 (avr32 architecture) with ap7000 processor and 32MB of SDRAM.

- (2.1 KB) Without XIP
- (2.1 KB) With XIP

Future Work

Here is a list of things that could be worked on for this feature:

```
-
```

Category:

- File Systems

From: eLinux.org

Pram Fs Specification

```
#noprint
VERSION 0.1 - for other versions, click the "info" button.

'''Table Of Contents:'''
[[TableOfContents]]
#noprint
```

Contents

Introduction

This page specifies a file system that will enhance security of system data in the presence of kernel bugs or rogue programs.

Rationale

A single bug in the Linux kernel may cause catastrophic damage to a system. If a product holds irreproducible security keys, financial data, or account information, then loss of such data could render the product unusable, or worse. The customer could suffer financial or legal harm (from account theft or identity theft).

It is not possible to guarantee with certainty that there are no bugs in the Linux kernel. However, it is possible to decrease the probability that a bug in the kernel will cause damage to a particular area of memory or storage. This protected area could then be used with greater confidence to hold sensitive user or product data.

Portions of the product memory and storage should be made resistant to kernel bugs. A protected RAM file system would remain consistent if any of the kernel data pointers are corrupted, or if the kernel starts executing unexpectedly in the wrong location.

Specifications

1. A configuration option for the Linux kernel SHALL be provided which controls whether or not the kernel supports the PRAM file system. This option MUST be called CONFIG_PRAM_FS. 2. A full-featured read/write file system for Linux that is RAM-based.
2. If the memory is non-volatile, the file system SHALL be persistent.
3. File I/O in PRAMFS is always direct, synchronous, and never blocks.
4. PRAMFS should be write-protected.
5. In case there are systems where the write protection is not possible, this feature can be disabled with the CONFIG_PRAMFS_NOWP configuration option.

Notes (informational and non-normative)

There is only minimal effort required to back-port the 2.4.22 version of the PRAMFS patch set to the CELF source tree.

References

Patches are available for PRAMFS against the kernel.org trees for kernel versions 2.4.22 and 2.6.4 at http://pramfs.sourceforge.net.

Remaining Issues

```
#noprint
[this is a placeholder section for listing issues while the spec is under development.
It should be empty when the spec is completed (or the issues should be deferrable to
a subsequent version of the spec).]
#noprint
```

Categories:

- File Systems
- Specification

> From: eLinux.org

Squash Fs

Contents

Description

"Squash FS" is the name of a compressed read-only filesystem for Linux. There are a number of such file systems available for Linux, including ROMFS, CramFS and SquashFS.

Rationale

A compressed file system is interesting in embedded systems for reducing the overall size (in flash) of the Linux system. Squash FS is reported to have better compression capabilities than CramFS, which is a very popular.

Resources

Projects

The SquashFS home page is at: Squash FS

Status/News

SquashFS version 4.0 was accepted into the mainline kernel in January

1. See http://lwn.net/Articles/314326/ and the kernel change log at: http://kernel.org/pub/linux/kernel/v2.6/testing/ChangeLog-2.6.29-rc1

As of November 2009, Phillip Lougher was working on adding LZMA support to the file system. See http://old.nabble.com/Re%3A-squashfs-4.0-lzma-support-td25132198.html#a26028786

Downloads

Patch

- See the Squash fs download page for patches
- if you want LZMA support, read below

Utility programs

The squashfs file release contains a README, the squashfs patch files, and the squashfs-tools directory (mksquashfs). Please see the INSTALL file for install instructions.

How To Use

See the Squash FS Howto:

- online version
- Squash FS Howto page (copy of document in this wiki - may be out of date)

Sample Results

Here are brief summaries for 2 large file systems, saved using a variety of file system types.

This information was provided by Phillip Lougher.

Ubuntu liveCD compression results

```
ext3 uncompressed size      1.4 GB
ISO9660 uncompressed size   1.3 GB
Zisofs compressed size      589.81 MB
Cloop compressed size       471.89 MB
Squashfs2.0 compressed size 448.58 MB
Squashfs2.1 compressed size 448.58 MB
```

Damn Small Linux liveCD compression results

```
ext3 uncompressed size      126 MB
CRAMFS compressed size      52.19 MB
Squashfs2.0 compressed size 46.52 MB
Squashfs2.1 compressed size 46.52 MB
```

File System Performance

There is performance data for these file systems on the page Squash Fs Comparisons

LZMA

LZMA is the name of a compression algorithm that can be used to achieve better compression with SquashFS.

Benefits of LZMA:

- Better compression - easily saving 5% on a file system

Disadvantages:

- Decompression is slower than zlib

Availability

Up until Fall of 2009, support for the LZMA compression algorithm in SquashFS was available via a set of external patches. See http://www.squashfs-lzma.org/

In December of 2009, Phillip Lougher submitted some patches to LKML for LZMA support in SquashFS. See http://lkml.org/lkml/2009/12/7/84

mksquashfs instructions

(As of December, 2009...)

To obtain a mksquashfs that supports LZMA, see the Squashfs CVS (http://sourceforge.net/projects/squashfs/develop).

You'll need to edit the squashfs-tools Makefile to enable LZMA support. The comments in the Makefile should, hopefully, explain how to build LZMA support into Mksquashfs/Unsquashfs.

Once built-in, LZMA support can be specified using the -comp lzma option, i.e.

```
mksquashfs dir dir.img -comp lzma
```

Unsquashfs doesn't need any extra options, it automatically detects which compression has been used.

You can tell which compression algorithms Mksquashfs/Unsquashfs support by just typing the command on its own (i.e. mksquashfs, or unsquashfs). The (de-)compressors available are displayed at the end of the output.

lzma data comparison

Here is some information that was posted recently (?) on the squashfs mailing list by Oleg Vdovikin:

```
> # du -s target
> 7836    target
> # ls -l target.*
> -rw-r--r--  1 root     root 2842788 Aug 27 17:54 target.cramfs
> -rwx------  1 root     root 2449408 Jan 26 13:19 target.sqshfs
> -rwx------  1 root     root 2060288 Jan 26 13:21 target.lzmafs
>
>     So, lzma for this filesystem gives 84% of original size. For bigger
> filesystem I've got 82%.
```

Notes on compression

```
From: John Richard Moser <nigele...@comcast.net>
Date: Wed, 29 Sep 2004 18:20:11 +0200
Subject: Re: Compressed filesystems: Better compression?

Matti Aarnio wrote:
| Compression algorithms are a bit tough to be used in a random access
| smallish blocks environments. In long streams where you can use megabytes
| worth of buffer spaces there is no problem is achieving good performance.
| But do try to do that in an environment where your maximum block size
| is, say: 4 kB, and you have to start afresh at every block boundary.

Yes of course. I've seen the compressed page cache patch do this and
get fair peformance (10-20%), though on double size blocks (8KiB) it
manages almost twice as good (20-50%, averaged around 30% IIRC). Not
great, but not bad.

On compressed filesystems you can work with 64k or 128k blocks.
Somewhere around 32-64k is usually optimal; you're not going to see
great improvements using 1M blocks instead of 512k blocks.

| Whatever algorithms you use, there will always be data sequences that
| are of maximum entropy, and won't compress. Rather they will be
| presented in streams as is with a few bytes long wrappers around
| them.

Yes, an intelligent algorithm decides that if the underlying compression
algorithm used produces no results, it just marks the block as
uncompressed and stores it as such. ZLIB does this if the block gets
bigger. LZMA might not; but higher level intrinsics (block headers)
could handle that easy (as you said).
```

Category:

- File Systems

> From: eLinux.org

UBIFS

Contents

Introduction

UBIFS is a filesystem that works on top of UBI volumes

- the UBIFS home page is at: http://www.linux-mtd.infradead.org/doc/ubifs.html
- UBIFS presentation slides: ubifs.odp

It represents a next-generation flash filesystem, compared to JFFS2, and (via UBI) operates on raw flash rather than on block-based media (such as eMMC or traditional hard drives).

History

UBIFS was written by developers at Nokia with help from the University of Szeged. Included in these developers was the current (as of 2012) maintainer of UBIFS, Artem Bityuski. The filesystem was mainlined in the Linux kernel in July, 2008, in kernel version 2.6.27.

Features

The UBIFS home page has more detailed information, but in summary, UBI has the following features relative to other flash filesystems:

- UBI/UBIFS scales to large flash sizes better than JFFS2
- Good fault tolerance, via a number of features
- Built-in on-the-fly compression
- Good runtime performance

Fastmap support

As of kernel version 3.7, UBI fastmap support was added to the kernel, to overcome a scalability issue with the time required to mount UBIFS on large flash media.

I specifically deals with the scalability issue described here: http://www.linux-mtd.infradead.org/doc/ubifs.html#L_scalability

A description of the feature is at: http://lwn.net/Articles/517422/

```
"UBI Fastmap is an optional feature which stores the physical to
logical eraseblock relations in a checkpoint (called fastmap) to reduce
the initialization time of UBI. The current init time of UBI is
proportional to the number of physical erase blocks on the FLASH
device. With fastmap enabled the scan time is limited to a fixed
number of blocks."
```

UBIFS vs. YAFFS2 comparisons for 2.6.31.1

Note: see our Flash_Filesystem_Benchmarks for more recent benchmarks.

Hardware: MIPS, 403MHz CPU, 1GB Nand Flash

IOZone results: 4M, 8M & 16M file sizes in 980MB partition.

- mount time
  - "1st mount" : time for mounting just after "flash_eraseall".
  - "Empty" : time for mounting after "1st mount".(there's no files in partition)
  - "Full" : time for mounting after creating files util the partition is full.(file size is random.)
  - "Ubiattach" time for attaching 980MB partition into the ubi layer using ubiattach util.

	Ubiattach	Ubifs	Yaffs2
1st mount	2.59 sec	0.17 sec	2.25 sec
Empty	2.57 sec	0.11 sec	0.03 sec
Full	2.63 sec	0.16 sec	0.80 sec

- IOZone results

UBIFS compared to YAFFS2 (file size = 4MB)

4M	write	rewrite	read	reread	r.read	r.write	b.read	r.rewrite	s.r
yaffs2	1538	1527	296962	297696	297825	1521	296876	1526	29
ubifs	19913	20989	298346	297631	299809	20968	298368	20181	29
ubifs/yaffs2	12.95	13.75	1.00	1.00	1.01	13.79	1.01	13.22	1.0

UBIFS compared to YAFFS2 (file size = 16MB)

16M	write	rewrite	read	reread	r.read	r.write	b.read	r.rewrite	s.r
yaffs2	1538	1523	297199	298525	298896	1528	298433	1525	29
ubifs	20501	20656	298272	299109	299032	20521	298417	20710	29
ubifs/yaffs2	13.33	13.56	1.00	1.00	1.00	13.43	1.00	13.58	1.0

UBIFS compared to YAFFS2 (file size = 32MB)

32M	write	rewrite	read	reread	r.read	r.write	b.read	r.rewrite	s.r
yaffs2	1539	1525	296537	297204	297253	1527	297177	1527	29
ubifs	20474	20611	296765	297490	297334	20659	296972	38416	29
ubifs/yaffs2	13.30	13.52	1.00	1.00	1.00	13.53	1.00	25.16	1.0

Creating UBI Image

This is easiest to do, if you have access to the device and can run ubinfo and dmesg, otherwise you'll need to determine the volume size, Logical Erase Block size, etc by other means. UBI has some block overhead, which I found documentation inconsistent with my particular application, so your results may very. If your device is one UBI image for the entire NAND, this should be easier, and could probably be determined by just mounting a copy of the UBI image from the device if available.

To create the image from a rootfs you've built first you need to create the ubi.ini file, that describes your ubi image. Create a regular text file, ubi.ini, example contents, for more info run ubinize -h:

```
[ubi_rfs]
mode=ubi
image=ubifs.img
vol_id=0
vol_size=87349248
vol_type=dynamic
vol_name=ubi_rfs
vol_alignment=1
vol_flags=autoresize
```

Next you'll run the commands that actually build it. Here ubi.ini is the file you just created, ubifs.img is a temp file you can delete once you are done, and your_erootfs.ubi is the name of the rootfs image that will be created.

```
sudo /usr/sbin/mkfs.ubifs -m 2048 -e 129024 -c 677 -r /path/to/rootfs ubifs.img
sudo /usr/sbin/ubinize -o your_erootfs.ubi -p 131072 -m 2048 -s 512 -O 512 ubi.ini
```

To determine these and the ubi.ini file settings, use ubinfo -a and dmesg on the device if possible, which both give plenty of information about the values needed. The size and vol_name are listed under "Present volumes" when you run ubinfo -a on the device. The second half of that particular ubi device's description. While the NAND description's PEB, LEB etc are in dmesg.

mkfs.ubifs

- -m - Minimum I/O unit size.
- -e - Logical Erase Block (LEB) size.
- -c - Max LEB count. (vol_size/LEB)
- -x - Compression type: lzo (default), favor_lzo, zlib, none
- -r - Path to root filesystem.
- ubifs.img - Temporary image file.

ubinize

- -o - Output file.
- -p - Physical Erase Block (PEB) size.
- -m - Minimum I/O unit size.
- -s - Minimum I/O size for UBI headers, eg. sub-page size.
- -O - VID header offset from start of PEB.
- ubi.ini - UBI image configuration file.

Mounting UBI Image on PC using nandsim

First create a simulated NAND device (this one is 256MB, 2048 page size). _id_byte= corresponds to the ID bytes sent back from the NAND.

```
$ sudo modprobe nandsim first_id_byte=0x20 second_id_byte=0xaa third_id_byte=0x00 fourth_id_byte=0x15
```

Check it was created.

```
$ cat /proc/mtd
dev:    size   erasesize  name
mtd0: 10000000 00020000 "NAND simulator partition 0"
```

Next, attach it to a mtd device.

```
$ sudo modprobe ubi mtd=0
```

I had to detach it prior to formatting it.

```
$ sudo ubidetach /dev/ubi_ctrl -m 0
```

If that ubidetach step fails when you enter it, just proceed to the next step to format the mtd device.

```
$ sudo ubiformat /dev/mtd0 -f <image>.ubi
ubiformat: mtd0 (nand), size 268435456 bytes (256.0 MiB), 2048 eraseblocks of 131072 bytes (128.0 KiB), min. I/O size
 2048 bytes
libscan: scanning eraseblock 2047 -- 100 % complete
ubiformat: 2048 eraseblocks have valid erase counter, mean value is 1
ubiformat: flashing eraseblock 455 -- 100 % complete
ubiformat: formatting eraseblock 2047 -- 100 % complete
```

Then, attach it.

```
$ sudo ubiattach /dev/ubi_ctrl -m 0
UBI device number 0, total 2048 LEBs (264241152 bytes, 252.0 MiB), available 0 LEBs (0 bytes), LEB size 129024 bytes
(126.0 KiB)
```

Make a target directory, and mount the device.

```
$ mkdir temp
$ sudo mount -t ubifs ubi0 temp
```

> From: eLinux.org

Power Management

Contents

Introduction

This page has information about Power Management for Linux. Power Management exists because many products are handheld or mobile, and consumers are interested in using their products for as long as possible on a single battery charge.

Power Management Technology/Project pages

- http://www.lesswatts.org/index.php
  - LessWatts.org
  - LessWatts.org is about how you can save real watts, however you use Linux on your computer or computers.
  - LessWatts is about creating a community around saving power on Linux, bringing developers, users, and sysadmins together to share software, optimizations, and tips and tricks.
  - LessWatts also provides the powertop tool, which helps to identify some power hogs.
- OMAP Power Management
  - For Power Management on processors in the Texas Instruments OMAP family.

Linux Power Management Mini-Summit

Mini-Summit Notes

- 2010 Notes from the Boston Linux Power Management Mini-Summit
- 2009 Notes from the Montreal Linux Power Management Mini-Summit
- 2008 Notes from the Ottawa Linux Power Management Summit

CE Linux Forum Standards

See here CELF PM Requirements 2006

Documents

- Device_Power_Management_Specification

- Dynamic_Power_Management_Specification

- Mapping of ACPI states to omap power states: ACPI to OMAP2 Mapping

For some good overviews of different PM features relevant to embedded, you may want to look at the following papers:

- Every Microamp is Sacred - A Dynamic Voltage and Current Control Interface for the Linux Kernel - Liam Girdwood Slides and video
- Power Management Quality of Service and How You Could Use it in Your Embedded Application - Mark Gross Slides and video
- Building Blocks for Embedded Power Management - Kevin Hilman Slides and video
- Linux Suspend-to-Disk Objectives for Consumer Electronic Devices - Vitaly Wool Slides.
- Linux Clock Management Framework - Siarhei Yermalayeu slides
- Advanced Power Management for OMAP3, Peter de Schrijver, FOSDEM 2009 Video
- Taking Linux power management to production quality, Eugeny Mints, ELCE 2008 Video.
- Power Management on ARM11, Mischa Jonker, ELCE 2008 Slides and video.
- Power_Management_Specification
- Static_Power_Management_Specification

Open Source Projects/Mailing Lists

- linux-pm mailing list (and list archives).
- Dynamic Power at sourceforge.

Category:

- Power Management

> From: eLinux.org

Device Power Management Specification

Contents

Background

Various devices on embedded platforms upport low-power states that can be employed by CE products at times when full operation of the device is not required. The ability to manage device power usage may be crucial to many CE products, especially those powered by batteries.

Generic Linux contains some support for these topics. However, device suspend/resume support is not a priority at present, and neither is support for embedded platforms. This specification addresses the potential lack of device power management features on a platform used for CE products by requiring that a CELF-conforming Linux for that platform support its basic device power management capabilities. This specification also requires a minimal set of functionality closely associated with device power management. The basic capabilities outlined here may be extended by future CELF specifications that cover additional features useful for CE products.

Terminology

CE platform device : A device that is closely associated with the platform, that is supported by a Linux driver under an open-source license, and that may reasonably be expected to appear in an actual consumer electronics product based on the platform. See further discussion below.

Device resume : The process of restoring the state of a device that was previously suspended to normal operation.

Device suspend : The process of placing a device into a device suspend state, and/or of preparing the device for a system suspend. This may occur for such reasons as explicit instructions to power down an unneeded device by an application, or as part of a system suspend. Depending on the platform and device, device suspend may include saving state to allow later restoration of state at device resume time.

Device suspend state : A reduced-power state supported by a device. Many, but not all, devices support at least one device suspend state, which may prevent operation of the device until a device resume is performed. The various device suspend states may be activated by device-specific interfaces and/or automatic criteria, or may follow standards such as ACPI. This specification primarily targets reduced-power states that are activated by the device driver when needed (rather than automatic hardware mechanisms, such as inactivity timers).

Device Suspend/Resume Discussion

Device suspend may occur for reasons that include:

-
  - an application explicitly manipulates device state, such as to power down a device no longer required by the application
  - a system suspend occurs, which suspends all devices prior to suspending the system
  - a power policy management subsystem, such as DPM, places the system in a state that is incompatible with

operation of the device
    - a hardware or software mechanism triggers a low-power state after a period of inactivity
    - the driver powers down the device because applications no longer hold an open reference to the device

In many cases, CELF specifies support for evaluation or reference boards, based on which CE products may be derived using a custom hardware design that incorporates the processor and various devices. This specification targets only devices and drivers termed "CE platform devices" here, which meet these criteria:

- The device is closely associated with the platform. A CE platform device may be physically located on a single-board computer or in some other way be tightly coupled to the platform supported by CELF-conforming kernel source, such that its presence is likely in many products that may be based on the platform. Devices not included in this definition include arbitrary cards that plug into buses provided on the platform, such as PCI or PCMCIA, or that may be attached in custom hardware designs.
- The device is supported by a Linux driver under an open-source license.
- The device may reasonably be expected to appear in an actual consumer electronics product based on the platform. A CE platform device may be distinguished from devices that are present on evaluation or reference boards for development or debugging purposes, such as an ethernet interface that is unlikely to appear in an actual CE product.

This specification targets CE platform devices exclusively, in order to give product designers the necessary tools to save power in actual product configurations. This distinction is made in order to avoid mandating power management capabilities for:

- devices not present in the CELF-supported evaluation/reference boards
- devices for which no open source driver has been made available
- devices that serve only a development or diagnostic function

Device state may need to be be saved during the device suspend operation, such that device operation can later be restored to approximately the same condition at device resume. If so, device state is typically saved in SDRAM since SDRAM is usually powered (perhaps in self-refresh state) during the suspend interval -- a platform that does not preserve SDRAM during suspend generally must reboot at resume time, whereupon device state can be restored from stable storage if needed.

System suspend may remove power from some or all devices, depending on the platform and the particular system suspend state entered, leaving the devices unpowered during the suspend interval and restoring power at system resume. This may affect the manner in which device resume occurs during system resume, since recovering from a power cycle may require different procedures than are needed for individual device suspend/resume. This may also affect the actions to be taken to accomplish a device suspend during system suspend; for example, entering a low-power suspend state may not be useful if the platform is about to remove power from the device.

Specification

1. If a CE platform device that supports device suspend/resume actions, then both kernel programmatic interfaces and userspace interfaces MUST be provided to individually perform device suspend and device resume for only that device. These interfaces SHOULD suspend other devices that depend upon the selected device for correct operation. Other devices that do not depend upon the selected device for correct operation MUST NOT be suspended or resumed by these interfaces.

2. If a CE platform device supports a device suspend state, or if any actions are needed in order to correctly resume device operation after a system resume from at least one system suspend state supported by the Linux kernel, then the driver for the device SHOULD implement the support necessary for the device suspend and resume interfaces.

3. If a CE platform device supports one device suspend state, then the device suspend processing performed by the driver for the device MUST be capable of causing the device to enter the supported device suspend state. Where multiple device suspend states are available, the driver SHOULD be capable of entering each of these states.

4. The device resume processing performed by the driver for a CE platform device SHOULD restore device operation to approximate pre- suspend conditions.

5. A mechanism for requesting device suspend for all active CE platform devices at system suspend time MUST be provided. A mechanism for system resume to restore to an active state all CE platform devices that were in an active state prior to system suspend MUST be provided.

6. It is RECOMMENDED that drivers place devices into low-power states when not in use or after a period of inactivity.

7. It is RECOMMENDED that drivers and platform support code make use of hardware features to automatically place devices into lower-power states, such as to stop clocks (sometimes referred to as "automatic clock gating"), after a period of inactivity or when the hardware is in some manner able to detect that the device is not in use.

Non-normative notes

Among the choices for device suspend/resume interfaces are:

-
    -
        - The Linux 2.5/2.6 Linux Driver Model (LDM) implements a kernel API for driver suspend and resume functions. LDM also provides a kernel API for calling drivers to suspend all devices at system suspend, and to restore normal operation of the devices at system resume.
        - The Linux 2.5/2.6 sysfs filesystem exports interfaces that applications may use to suspend and resume devices (individually); these interfaces call the LDM driver suspend and resume kernel APIs.
        - APM function calls (such as pm_register) for device suspend/resume. Note that these interfaces are generally being replaced with Linux Driver Model interfaces in future Linux versions.

A version of the Linux 2.5/2.6 technology described above has been backported to Linux 2.4 for use in CELF-conforming systems based on the 2.4 Linux kernel.

Comments on the 4.5.2 specification section are the following:

- 4.5.2.4 and 4.5.2.6 does not make specific requirements upon devices that may change state in significant ways between device suspend and device resume, for example, a controller for a hotpluggable bus such as PCMCIA or MMC, where the consumer may insert or eject cards during the suspend interval.

- 4.5.2.5: In 2.6 kernel,Linux Driver Model (LDM) has this mechanism. CE platform devices that are already suspended via a previous individual device suspend should be skipped, if it is not necessary to perform any further device suspend processing prior to a system suspend (such as to save state in RAM in preparation for power-off). Devices that were already individually suspended prior to system suspend should be left in a suspended state after system resume.

- 4.5.2.6: For example, when the last open file descriptor for a device is closed, the driver may suspend the device.

Category:

- Specification

> From: eLinux.org

Dynamic Power Management Specification

Contents

Rationale

Many CE products are powered by batteries, not by a wired power supply. Making efficient use of the energy stored in the batteries is very important, both in terms of achieving an acceptable length of product usage per battery charge and in terms of product form factor; an energy-efficient product can be powered by a smaller sized battery than other would be required. This also has a bearing on the manufacturing cost of the product. Power savings can be achieved through the use of low-power system states, entered when the product is inactive or when selected explicitly by the user - for example, by use of the power switch to put the product into a standby state. This type of power management is referred to as static power management.

Dynamic power management refers to managing power while the product is in use, running user applications. Such power management can involve dynamic control of peripheral clocks and power supplies, varying the timer tick frequency during idle periods and CPU frequency/voltage scaling. This uses a combination of user-space and kernel-space software:

```
- a Policy Manager component running in user-space
- a Power Management Engine ("PM-Engine") component running in kernel-space
```

The software selects the system operating state based on a combination of the following:

```
- application requirements
- the parameters for battery lifetime or task deadlines
- CPU loading
- the current device constraints
- the current operating state
- user interaction
```

Specifications

If a platform has an interface to change its power parameters while platform is running, a CELF conforming kernel for the platform MUST satisfy the following:

1. A kernel SHOULD provide a userspace interface to add/remove/modify a power policy.
2. A kernel SHOULD provide interfaces to modify the platform's operating state and power parameters of the platform accordingly.
3. A kernel SHOULD have the capability to set the platform at the operating state of a task while the CPU is executing that task.
4. A kernel SHOULD have a userspace interface to initialize/remove/modify the operating state of a task.
5. A device driver framework MAY provide interface to register device constraints on device open and unregister the constraints on device close.

6. When a CPU is idle, a kernel SHOULD set the CPU at its low power mode if the CPU supports at least one low power mode.

7. System clocks MUST be correct regardless of power parameter changes.

Non-normative Notes

```
- Specific interface for an application to give hints on selecting an appropriate system operating state to the PM-En
gine is not required by this specification.
- Recommendations for Hardware Platform
 - The latency of changing power modes of a CPU should be small enough to do it whenever a CPU becomes idle.
 - Memory address/data buses and peripheral buses can be put to low power modes when CPU becomes idle if the latency
of putting those buses to low power modes are small enough.
 - CPU voltage/clock changing latency need to be quite small enough to do frequently while a task is running.
```

References

Power management framework - lower layer of PM-Engine:

1. IBM and Monta Vista Software, "Dynamic Power Management for Embedded Systems", Nov., 2002, http://www.research.ibm.com/arl/projects/papers/DPM_V1.1.pdf

2. "Dynamic Power Management", http://sourceforge.net/projects/dynamicpower/

Hooks required in tracking task scheduling etc.:

```
1.#3 "Kernel Hooks", http://www-124.ibm.com/developerworks/oss/linux/projects/kernelhooks/
```

Category:

- Specification

> From: eLinux.org

OMAP Power Management

Contents

PM branch

The PM branch is a developement branch of the linux-omap kernel for the purposes of developing and stabilizing the PM infrastructure for OMAP and submitting it upstream.

The maintainer of the PM branch is Kevin Hilman.

Features

- full-chip retention in idle and suspend
- full-chip OFF in idle and suspend
- idle PM via CPUidle
- support for multiple OMAP3/4 boards

The latest, tested PM branch is available as a branch named 'pm' from the linux-omap-pm repository. This branch is also sync'd daily as the 'pm' branch of the main linux-omap repository.

Current version

Supported platforms (OMAP3 only)

Tested on the following platforms using omap3_pm_defconfig with busybox-based initramfs, and tested full-chip RET and OFF in idle and suspend:

- 3430SDP
- OMAP3EVM
- Beagle

- Overo (Water + Tobi)
- Nokia N900 (a.k.a RX51)
- Zoom2
- KwikByte KBOC

Using OMAP PM

Features

By default, the OMAP is configured to hit full-chip retention in suspend.

Suspend/Resume

```
# echo mem > /sys/power/state
```

Serial console activity or other configured wakeup sources (keypad, touchscreen) will trigger resume.

Upon resume, you can use the powerdomain state statistics to check whether all states hit the desired state, cf. 'Debug info'

```
# cat /debug/pm_debug/count
```

In addition, if any power domains did not hit the desired state, you will see a message on the console.

Enabling system for hitting retention during idle

By default, the UARTs will not automatically idle when unused so they will prevent low-power states during idle. To enable UART idle timeouts with a 5 second timeout:

```
# echo 5 > /sys/devices/platform/omap/omap_uart.0/sleep_timeout
# echo 5 > /sys/devices/platform/omap/omap_uart.1/sleep_timeout
# echo 5 > /sys/devices/platform/omap/omap_uart.2/sleep_timeout
# echo 5 > /sys/devices/platform/omap/omap_uart.3/sleep_timeout
```

NOTE: the 4th UART is only present on 3630 and OMAP4.

Then, wait for any inactivity timers to expire (such as the 5 second UART timer) and check the powerdomain transition statistics to see that transitions are happening

```
 # cat /debug/pm_debug/count
```

Enabling system for hitting OFF

By default, retention is the deepest sleep state attempted. To enable power domain transitions to off mode

```
# echo 1 > /debug/pm_debug/enable_off_mode
```

Once again, after a suspend or after some idle time, use the power domain transition stats to check that transitions to off-mode are happening

```
 # cat /debug/pm_debug/count
```

Known Problems

- Zoom2/3: serial console wakeups not working
  - Problem: on suspend, by default the serial driver will disable serial interrupts, thus disabling the GPIO IRQ needed

for wakeup.
  - Fix: enable the wakeup feature for the tty used as console:

```
# echo enabled > /sys/devices/platform/serial8250.0/tty/ttyS0/power/wakeup
```

- GPIO module-level wakeups not always working
  - Background: GPIO wakeups can happen either via the GPIO module itself (module-level wakeups) or via IO pad wakeups if the CORE powerdomain is inactive, in retention or off.
  - If the IO pad wakeups are not enabled (either because CORE remains on, or because IO pad is not armed) GPIO wakeups may not happen unless the GPIO module-level wakeups are programmed correctly.
  - To ensure GPIO module wakeups are programmed correctly:
    - Enable GPIO IRQ for wakeup GPIO, including ISR. Use `request_irq()`
    - Ensure GPIO is edge-triggered. Only edge triggered GPIOs are wakeup capable (c.f. omap34xx TRM Sec. 25.5.3.1)
      - the `flags` argument of `request_irq()` should have either `IRQF_TRIGGER_FALLING` , `IRQF_TRIGGER_RISING` or both.
    - Enable GPIO IRQ as wakeup source using `enable_irq_wake(gpio_to_irq(<gpio>))`
  - **NOTE:** It is very important that an interrupt handler be configured for the GPIO IRQ, even if it does nothing but return `IRQ_HANDLED` . This is because without an interrupt handler, the GPIO IRQ event will never be properly cleared and this can prevent the GPIO module from hitting retention or off on the next idle request (c.f. omap34xx TRM Sec. 25.5.3.1).
- GPIO wakeup works once, but prevents future retention
  - See **NOTE** just above

Advanced features for PM developers and power users

Debug info

First, mount the debug filesystem (debugfs)

```
# mount -t debugfs debugfs /debug
```

Show powerdomain state statistics and clockdomain active clocks

```
# cat /debug/pm_debug/count
```

This will look something like this on OMAP3:

```
 # cat /debug/pm_debug/count
 cefuse_pwrdm (OFF),OFF:1,RET:0,INA:0,ON:0,RET-LOGIC-OFF:0
 always_on_core_pwrdm (OFF),OFF:1,RET:0,INA:0,ON:0,RET-LOGIC-OFF:0
 l4per_pwrdm (ON),OFF:0,RET:0,INA:0,ON:1,RET-LOGIC-OFF:0,RET-MEMBANK1-OFF:0,RET-MEMBANK2-OFF:0
 l3init_pwrdm (RET),OFF:0,RET:1,INA:0,ON:1,RET-LOGIC-OFF:0,RET-MEMBANK1-OFF:0
 cam_pwrdm (OFF),OFF:1,RET:0,INA:0,ON:0,RET-LOGIC-OFF:0,RET-MEMBANK1-OFF:0
 ivahd_pwrdm (RET),OFF:1,RET:1,INA:0,ON:1,RET-LOGIC-OFF:0,RET-MEMBANK1-OFF:0,RET-MEMBANK2-OFF:0,RET-MEMBANK3-OFF:0,RE
T-MEMBANK4-OFF:0
 mpu_pwrdm (ON),OFF:0,RET:0,INA:0,ON:1,RET-LOGIC-OFF:0,RET-MEMBANK1-OFF:0,RET-MEMBANK2-OFF:0
 cpu1_pwrdm (ON),OFF:0,RET:0,INA:0,ON:1,RET-LOGIC-OFF:0,RET-MEMBANK1-OFF:0
 cpu0_pwrdm (ON),OFF:0,RET:0,INA:0,ON:1,RET-LOGIC-OFF:0,RET-MEMBANK1-OFF:0
 tesla_pwrdm (RET),OFF:1,RET:1,INA:0,ON:0,RET-LOGIC-OFF:0,RET-MEMBANK1-OFF:0,RET-MEMBANK2-OFF:0,RET-MEMBANK3-OFF:0
 dss_pwrdm (RET),OFF:0,RET:1,INA:0,ON:1,RET-LOGIC-OFF:0,RET-MEMBANK1-OFF:0
 abe_pwrdm (ON),OFF:1,RET:0,INA:0,ON:1,RET-LOGIC-OFF:0,RET-MEMBANK1-OFF:0,RET-MEMBANK2-OFF:0
 gfx_pwrdm (OFF),OFF:2,RET:0,INA:0,ON:1,RET-LOGIC-OFF:0,RET-MEMBANK1-OFF:0
 core_pwrdm (ON),OFF:0,RET:0,INA:0,ON:1,RET-LOGIC-OFF:0,RET-MEMBANK1-OFF:0,RET-MEMBANK2-OFF:0,RET-MEMBANK3-OFF:0,RET-
MEMBANK4-OFF:0,RET-MEMBANK5-OFF:0
```

If you see each power domain has counters specified. OFF, RET, INA and so on...The count basically keeps incrementing every time it hits low power state. In the above example, cam_pwrdm (camera power domain) has hit OFF state once. GFX power domain has hit OFF state twice and like wise.

UART wakeup and timeout options

By default, each of the on-chip OMAP UARTs are enabled as wakeup sources. In addition, they are configured with a configurable inactivity timer (default 5 seconds) after which the UART clocks are allowed to be gated during idle or suspend.

For example, to disable the wakeup capability of a UART1 (a.k.a ttyO0)

```
# echo disabled > /sys/devices/platform/omap/omap-hsuart.0/power/wakeup
```

And to change the inactivity timer to 10 seconds, instead of the default 5:

```
# echo 10 > /sys/devices/platform/omap/omap-hsuart.0/sleep_timeout
```

Note that you can `cat` these files under `/sys` as well to see the current values.

UART PM Debugging Techniques

Debugging problems with the OMAP UART driver wakeup and data transfer when Power Management is enabled can be quite tedious, if one does not have a proper HW setup. An example of a setup (including both HW and SW changes) can be found in the OMAP_UART_pm_debugging page.

Public Power management test framework

Some commonly used power management utilities are listed here which make sense from an OMAP perspective

Cpufreq utils

cpufreq utils for testing dynamic voltage and frequency scaling.

Maemo pm_test

pm-test plugin for Maemo says

```
utility which tests that kernel and kernel modules works power management wise
```

This utility could be used to sanity test the powermanagement impact to a system for suspend/restore and basic power features.

Quick verification of suspend-idle functionality

the following script may be used with userspace supporting something simple as busybox:

```
#!/bin/ash
# Quick script to verify SUSPEND Resume behavior without human intervention
# Refer: http://elinux.org/OMAP_Power_Management for details

# Some params that might change based on the environment
SYS=/sys
DEBUG=$SYS/kernel/debug
PROC=/proc

PMDEBUG=$DEBUG/pm_debug
```

```
VOLTAGE_OFF=$PMDEBUG/voltage_off_mode
kver=`uname -r`
if [ $kver > "2.6.36" ]; then
    UART="$SYS/devices/platform/omap/omap-hsuart"
else
    UART="$SYS/devices/platform/serial8250"
fi
UART1=$UART.0/sleep_timeout
UART2=$UART.1/sleep_timeout
UART3=$UART.2/sleep_timeout

# Setup cpu idle
cpu_idle(){
    echo -n "$1" > $PMDEBUG/sleep_while_idle
}

# setup off mode
off_mode(){
    echo -n "$1" > $PMDEBUG/enable_off_mode
}

# Do a suspend
suspend_me(){
    echo -n "mem" > $SYS/power/state
}

# get my core data (This is the last domain to hit lowest power state)
core_count(){
    cat $PMDEBUG/count |grep "^core_pwrdm"
}

# get my retention counter
core_ret_count(){
    core_count|cut -d ',' -f3|cut -d ':' -f2
}

# get my off counter
core_off_count(){
    core_count|cut -d ',' -f2|cut -d ':' -f2
}

# setup wakeup timer - automated testing
wakeup_timer(){
    echo -n "$1" > $PMDEBUG/wakeup_timer_seconds
    echo -n "$2" > $PMDEBUG/wakeup_timer_milliseconds
}

# Setup our uart to be inactivity timer
setup_tty_sleep_timeout() {
    if [ -f $UART1 ]; then
        echo -n "$1" > $UART1
    fi
    if [ -f $UART2 ]; then
        echo -n "$1" > $UART1
    fi
    if [ -f $UART3 ]; then
        echo -n "$1" > $UART3
    fi

}

# Measurement Start
measure_start(){
    OFF_START=`core_off_count`
    RET_START=`core_ret_count`
    TIME_START=`date "+%s"`
}
# Measurement End
measure_end(){
    OFF_END=`core_off_count`
    RET_END=`core_ret_count`
    TIME_END=`date "+%s"`
```

```
}
# Common formatted print
measure_print(){
    DUR=`expr $TIME_END - $TIME_START`
    echo "$1 | $2 | OFF: $OFF_START->$OFF_END| RET:$RET_START->$RET_END ($DUR sec)"
}

# verify function
check_core_off(){
    RESULT=FAIL
    if [ $OFF_START -lt $OFF_END ]; then
        RESULT=PASS
    fi
}
check_core_ret(){
    RESULT=FAIL
    if [ $RET_START -lt $RET_END ]; then
        RESULT=PASS
    fi
}

# Disable everything
disable_all(){
    # disable voltage off
    if [ -f $VOLTAGE_OFF ]; then
        echo -n "0" >$VOLTAGE_OFF
    fi
    setup_tty_sleep_timeout 0
    wakeup_timer 0 0
    off_mode 0
    cpu_idle 0
}

# test idle - core ret
test_idle_ret() {
    disable_all
    measure_start
    setup_tty_sleep_timeout 5
    cpu_idle 1
    sleep 20
    disable_all
    sleep 1;sync
    measure_end
    check_core_ret
    measure_print "IDLE:RET test" $RESULT
}

# test idle - core off
test_idle_off() {
    disable_all
    measure_start
    setup_tty_sleep_timeout 5
    off_mode 1
    cpu_idle 1
    sleep 20
    disable_all
    sleep 1;sync
    measure_end
    check_core_off
    measure_print "IDLE:OFF test" $RESULT
}

# test suspend - core ret
test_suspend_ret() {
    disable_all
    measure_start
    wakeup_timer 5 0
    suspend_me
    disable_all
    sleep 1;sync
    measure_end
    check_core_ret
```

```
    measure_print "SUSPEND:RET test" $RESULT
}

# test suspend - core off
test_suspend_off() {
    disable_all
    measure_start
    off_mode 1
    wakeup_timer 5 0
    suspend_me
    disable_all
    sleep 1;sync
    measure_end
    check_core_off
    measure_print "SUSPEND:OFF test" $RESULT
}

# mount up the basic fs
already_mntd=`mount|grep $PROC`
if [ x == x"$already_mntd" ]; then
    mount -t proc none $PROC
fi

already_mntd=`mount|grep $SYS`
if [ x == x"$already_mntd" ]; then
    mount -t sysfs none $SYS
fi
already_mntd=`mount|grep $DEBUG`
if [ x == x"$already_mntd" ]; then
    mount -t debugfs none $DEBUG
fi
# Lets run the tests one by one..
NR=""
R=`test_suspend_off`
echo $R
NR="$NR\n$R"
R=`test_suspend_ret`
echo $R
NR="$NR\n$R"
R=`test_idle_off`
echo $R
NR="$NR\n$R"
R=`test_idle_ret`
echo $R
NR="$NR\n$R"
# Print End result summary
cat $PMDEBUG/count

# Print test summary
echo -e "$NR"
```

Categories:

- OMAP
- Power Management

> From: eLinux.org

Power Management Specification

```
#noprint
VERSION 1.0 - for other versions, click the "info" button.

'''Table Of Contents:'''
[[TableOfContents3]]
#noprint
```

Contents

Introduction

The goal of power management technology is to allow Consumer Electronic (CE) device manufacturers to optimize the power use of their products, including a variety of mobile devices, home appliances, and other end-user devices.

Rationale

In CE products, battery-driven mobile devices have particularly stringent requirements for power management, and this affects the product's competitiveness significantly in specific product categories. Mobile phones and other handheld devices are driven by secondary batteries, not wired power supply. Efficient power usage is required to increase the utilization time of such products.

Power Dissipation Elements in CMOS Circuits

CMOS(Complementary Metal Oxide Semiconductor) circuit has three elements of power dissipation; leakage current, dynamic current, short-circuit current power dissipation. The leakage current power dissipation is caused by the leakage current. The leakage current power dissipation is quadratically dependent on operating voltage but not dependent on clock frequency. The dynamic current power dissipation is caused by the dynamic current which is required for signal transitions. The dynamic power dissipation is quadratically dependent on the voltage and linearly dependent on the frequency of signal transitions. The short-circuit current dissipation is generated by short circuit current when NMOS and PMOS transistors are activated simultaneously. Short-circuit current power dissipation is either linearly or quadratically dependent on the supply voltage, depending on the size of the length of channel between PMOS and NMOS transistors. Of those three elements, the dynamic current power dissipation is a dominant factor because the frequency of signal transitions is very high.

Power Reduction Methods

There are various methods for reducing power consumption: voltage/clock scaling, clock gating, power supply gating, input signal selection and so on.

```
- The voltage/clock scaling is the method that reduces both the dynamic power dissipation and the static power dissip
ation by decreasing voltage and clock dependently or independently.
- The clock gating reduces the dynamic power dissipation by disconnecting the clock from an unused circuit block.
- The power supply gating disconnects the supply power from unused circuit blocks, which are components of whole circ
uit. In clock/power supply gating methods, it is important to decide whether the block is used or not and to restore
the blocks when they are used again.
- The input signal selection affects the reduction of the leakage power dissipation. The input signals of CPU pins sh
ould be adjusted to minimize the usage of power when the system is suspended.
```

Voltage scaling is the key to achieve power reduction and energy reduction. As clear from the analysis of the three power dissipation elements, operating voltage plays a major role in power dissipation. Let's take an example to see how voltage/clock scaling can reduce power and energy. If operating voltage is down-scaled from 2.0V to 1.0V, operating frequency is linearly dependent on the operating voltage in practice. So let us assume clock frequency at 1.0V is half of the frequency at 2.0V. In this case, nearly 87.5% of power can be reduced theoretically. In terms of energy, the total energy for finishing a job need to be compared. Please note that energy is the integration of power dissipation over a period of time. The time taken to finish a job at 1.0V is twice as that of 2.0V. So the total energy required to finish a job will be reduced by 75%.

http://tree.celinuxforum.org/docs/PM_formula.jpg

http://tree.celinuxforum.org/docs/PM_DVS.jpg

Clock gating and power supply gating can be used to reduce power usage by unused blocks. It is important to predict correctly the idle period of a hardware block because incorrect prediction may lead to performance degradation due to the latency of reconnecting clock or power to the hardware blocks.

Power Management Architecture

In designing power management architecture for Linux, the followings are kept in mind.

```
- To design a system-level optimization of power usage involving interactions among application, OS, and hardware.
- To provide a generic power management framework for various hardware platforms
```

System-level power management is important for coordinating the performance and power dissipation of hardware components in a system. Recently, CPU chip vendors deploy dynamic clock/voltage scaling support in their processors to enable OS-level power management capability. The power reduction techniques offered by hardware components should be synthesized, managed, and adjusted to minimize the power dissipation of the whole system while the requirements of system should be satisfied as well. Existing applications need to be adapted to use the power management capability of operating system and such applications are said to be "Energy-Aware".

The power management architecture is designed to provide a generic framework for various hardware architectures. Hardware specific power control interfaces are abstracted by a power control layer. Power management can be implemented for various hardware platforms by adapting the machine-specific power control layer as necessary.

http://tree.celinuxforum.org/docs/PM_Arch.jpg

In this specification, power management technologies are classified into three categories; platform suspend/resume, device power management, and platform dynamic power management. Platform suspend/resume is aimed at major power state changes at infrequent intervals, while platform dynamic power management is aimed at subtle changes across a wider range of power states at frequent intervals. Platform suspend/resume technology is used for reducing most of power dissipation of a product while the product is idle for a long time. In contrast, dynamic power management is used for

reducing power dissipation while a product is being used. Dynamic power management covers also reducing power for very short idle periods of time while a product is busy. Device power management is to suspend/resume devices in a platform, which is used by platform suspend/resume and dynamic power management technologies.

The rest of this specification is organized as follows. In Terminology section, general power management related terminologies are defined. Platform suspend/resume, device power management, and platform dynamic power management are described in separate sections. Work in Progress section describes important on-going projects which are not included in CELF 1.0 spec but need to be considered for next CELF specification.

Work in Progress

Platform suspend/resume

Protocol between Energy-Aware Application and Policy Manager

1.
   i. Energy-aware(EA) application SHOULD register itself to policy manager.
   ii. Policy manager SHOULD send system suspend request to all applications registered as EA application.
   iii. Energy-aware application SHOULD respond to the system suspend request from policy manager appropriately.
      i. If EA application agree to system suspend, it invokes its suspend handler.
         i. The suspend handler may store important data including user data to a non-volatile storage.
         ii. The suspend handler SHOULD send acknowledgement to policy manager and SHOULD wait until it receives resume request from policy manager.
      ii. If EA application disagree to system suspend, it SHOULD send negative acknowledgement to policy manager.
2.
   i. If system inactivity which is longer than system timeout threshold is detected, policy manager MUST send system suspend request to EA applications and wait for responses from EA applications.
      i. The policy manager SHOULD provide user interface for initializing the system timeout threshold.
      ii. The policy manager SHOULD initialize system timeout threshold to a default value.
      iii. The policy manager MAY check device inactivity by periodically reading last access time of device.
   ii. If a user has requested system suspend, policy manager SHOULD enforce every EA application to suspend itself and wait for responses from EA applications. An EA application which is enforced to suspend by policy manager SHOULD invoke its suspend handler unconditionally.

Device Power Management

Automatic Device Power Saving

1.
   i. The policy manager set the length of inactivity period for each device.
   ii. The period of device inactivity can be detected by PM-engine/device drivers and notified to policy manager via signal.

Dynamic Power Management

Energy-Aware(EA) Application Specification

This specification shall define requirements of energy-aware applications so that application can interact tightly with in-kernel PM-engine to control system power states dynamically adaptive to various application specific environments.

EA application may choose one appropriate operating state among system PM policy. The system policy may be given to the EA application by policy manager or hard-coded, which is dependent on design decisions by system administrator. If the system policy information should be given by policy manager, EA application need to set up a IPC mechanism with the policy manager while it starts up.

For real-time EA applications, the information regarding the real-time task scheduling such as the period of execution and their priorities should be given to in-kernel PM engine via some whatever existing API which we have to choose after extensive discussion on pros and cons of existing APIs. (Linux kernel need to support scheduling periodic real-time tasks.)

There are some cases that an EA application need to change system operating state according to the task's environment. This situation is highly dependent on system constraints which system administrator should know correctly. If a system administrator firmly believe that the operating state of an EA task need to be changed, the decision by system administrator should be respected and the in-kernel PM engine will proceed with the decision. It should be noted that system performance and power consumption efficiency might be greatly hampered if the decision of the system administrator is incorrect. So the system administrator should be very much considerate on changing application operating state.

Performance vs. Battery Lifetime Policy

1. A kernel SHOULD provide userspace interface for specifying requirements on the lower limit of battery lifetime. If the battery lifetime requirement exists, a kernel SHOULD set the operating state of the task to allocate the remaining energy resource amongst all tasks, based on the task priorities, to satisfy this requirement.

Category:

- Specification

> From: eLinux.org

Static Power Management Specification

Contents

Background

Embedded platforms typically offer one or more low-power system states in which power consumption is lowered through such means as stopping clocks and/or powering off some or all system components. CE products may take advantage of these states to save power during periods of inactivity by entering these states. This is particularly important crucial to CE products powered by batteries.

Generic Linux contains some support for these topics. However, the community emphasis is currently on certain features, primarily hibernate-to-disk, for laptop/notebook computers based on desktop-compatible processors (and powered by batteries much larger than are used in many CE products, such that aggressive power savings may not be as critical). Basic platform suspend/resume support as defined here is not a priority at present, and neither is support for embedded platforms. One of the largest efforts is for hardware based on the ACPI standard, which is not implemented on most embedded platforms.

This specification addresses the potential lack of static power management features on a platform used for CE products by requiring that a CELF-conforming Linux for that platform support its basic system suspend/resume capabilities. This specification also requires a minimal set of functionality closely associated with static power management. The basic suspend/resume capabilities outlined here may be extended by future CELF specifications that cover additional features useful for CE products.

Scope

This specification generally covers the basic functionality needed in an embedded linux system to save power by suspending and resuming operation of the system, utilizing suspend states implemented by the platform. The ability to suspend and resume systems is often referred to as "static power management" in the CELF Power Management Working Group.

System suspend is usually a relatively lengthy procedure, undertaken when an extended period of product inactivity is expected. This procedure can thus be distinguished from the power state changes controlled by more dynamic mechanisms, such as the IBM-MontaVista Dynamic Power Management subsystem (DPM), which may be executed more frequently during very brief idle periods, and are covered in separate CELF power management specifications. The system power state modifications made by dynamic power management schemes during brief idle periods typically result in less power savings than do the system suspend states covered in this specification. Furthermore, the dynamic power management state modifications normally have lower entry and exit latencies than do system suspend states, and require few or no interactions with device drivers. A system suspend, on the other hand, may take a noticeable amount of time to suspend and resume, and in many cases requires interactions with device drivers to save state and/or to enter and exit device suspend states.

A related topic not explicitly covered in this specification is "hibernate-to-disk", where system state is saved to a stable storage device (such as a hard disk) during a system suspend and restored from storage at system resume. The state saved usually includes most or all of RAM. This type of system suspend/resume is commonly implemented for laptop/notebook computers. Because the more specialized nature of hibernate-to-disk (which is typically a software feature not tied to the specifics of the base platform or devices), as well as the fact that this functionality is only appropriate for certain hardware and product configurations with large and fast storage devices available for saving state, hibernate-to-disk is not covered in this specification. This topic may be covered in future CELF power management specifications.

Platforms vary considerably in the kinds of support provided for static power management, and any attempt to standardize these technologies for a wide variety of platforms will necessarily be subject to a number of grey areas. The intent of this specification is to ensure a developer that, by choosing a CELF-conforming kernel source base for a particular platform, a number of commonly useful mechanisms and interfaces are available to take advantage of basic static power management features of that platform. In many cases this basic support will need to be augmented to obtain the exact behaviors desired for a particular product.

Terminology

The following terms are used in this portion of the specification:

Active state A system or device that is not in a suspend state is said to be in an "active state".

```
SDRAM self-refresh mode::: A platform capability that typically saves additional power during times in which no SDRAM
 access may occur.
```

System suspend The process of placing the system into a system suspend state. This may be triggered, for example, in response to the consumer pressing a "standby" button on the product. This process generally also performs a device suspend sequence for most or all devices.

System suspend state A reduced-power state supported by the platform. Modern embedded platforms often offer a variety of modifiable parameters and execution modes related to power consumption; this specification primarily targets those system suspend states used to dramatically reduce power consumption for extended periods of time (further clarification appears in the discussion below).

System resume The process of restoring the state of the system to approximate pre-suspend conditions when resuming operation after a previous system suspend. System resume may be triggered, for example, when the consumer presses a "wakeup" button or when the product is automatically awoken by an alarm or external event (such as incoming call on a phone). This process generally also performs a device resume sequence for most or all devices.

Wakeup method A platform mechanism for exiting a system suspend state. Wakeup may be triggered by such means as pressing a button, I/O activity (such as sending characters to a serial console device), or asserting an interrupt.

System Suspend/Resume Discussion

System suspend may be triggered for such reasons as:

- the consumer presses a "standby" button on the product
- the product automatically suspends after the product has not been in use for a certain period of time
- an application or kernel driver suspends the system due to such conditions as a low battery strength indication or temperature threshold exceeded

The documentation for the particular platform may refer to its system suspend states using names such as "standby", "sleep", or "suspend" states or modes. The precise characteristics of system suspend will vary by platform and by the particular system suspend state. A system suspend state might entail some or all of the following actions:

- stopping or slowing various clocks
- removing power from some or all devices, or placing devices into a low-power state
- placing SDRAM into self-refresh mode

- lowering core operating voltage and other platform power parameters
- halting CPU execution

This specification places no requirements upon the exact behavior of system or device suspend states, or upon the wakeup methods that resume normal system operation. The intent is to ensure that a CELF-conforming Linux supports the basic system and device suspend/resume features offered by the platform to a minimal degree.

This specification does require that a system suspend first call drivers for devices in an active state, in order to prepare for the system suspend. Depending on the platform and system suspend state characteristics, such preparation may be useful to further lower system power consumption by placing all appropriate devices into a suspend state, and/or may be necessary to save device state such that device operation can be later resumed during system resume. If the platform will remove power from the device during the system suspend state then first placing the device into a low-power state is, of course, not necessary, but it is commonly the case that device state will need to be saved in order to reconfigure the device when later re-powered.

If multiple system suspend states are implemented by the platform and supported by Linux, then a CE application may choose among the suspend states based on product-specific criteria such as whether a particular suspend state is compatible with proper functioning of required I/O devices during the suspend, the latency involved when entering and exiting the state, and so forth.

Specifications

1. If the platform supports at least one system-suspend state and at least one method to wake up from the system-suspend state, a configuration option for the Linux kernel SHALL be provided that controls whether the kernel has the ability to enter and wake up from at least one system suspend state.
2. If the platform supports more than one system suspend state, the ability to enter and wake up from all platform-supported system-suspend states SHOULD be implemented.
3. Both a kernel programmatic interface and a userspace interface to initiate system suspend SHOULD be provided.
4. If the platform supports SDRAM self-refresh mode and supports at least one system-suspend state that leaves SDRAM powered, the Linux kernel system-suspend code SHOULD have the ability to place SDRAM in self-refresh state.
5. The kernel SHOULD provide interfaces for delivering power management events, including power button, smart battery, and temperature sensor events, to user space.
6. A configuration option for the kernel SHOULD be provided that controls whether the kernel supports a hibernation technique by which the system preserves state on disk or flash memory during suspend state and restores the state from disk or flash memory upon resume.

Specific interfaces to perform the above are not required by this specification; see Non-Normative Notes for discussion.

Non-normative notes

Among the choices for system suspend/resume interfaces are:

- The Linux 2.5/2.6 PM subsystem implements a kernel API for system suspend and resume, with limited support for choosing a specific system suspend state.
- The Linux 2.5/2.6 sysfs filesystem exports interfaces that applications may use to suspend the system; these interfaces call the PM subsystem kernel API.
- The /proc/sys/pm interface for system suspend/resume. Note that these interfaces are generally being replaced with the PM subsystem in future Linux versions.
- The Dynamic Power Management subsystem (DPM) offers an interface for system suspend that may be convenient to use in systems that employ DPM for other power management tasks.
  - Hardware Platform Dependencies
    - The ability to support a special register used by boot-loader to detect whether the type of booting is cold boot or system resume.

  - Usually, platform is designed to support power supply gating which is to disconnect power from a part of PCB. The power supply gating can eliminate both leakage current and dynamic current while clock gating can eliminate dynamic current only. PCB has multiple power plane whose power can be gated independently.
  - Boot-loader need to detect whether booting is cold boot or system resume.
  - There are four types of power management events typically; button event, battery event, thermal event, and timer events. Those events are useful for predictable power control and prevent a system from being overheated.

A version of the Linux 2.5/2.6 technology described above has been backported to Linux 2.4 for use in CELF-conforming systems based on the 2.4 Linux kernel.

Category:

- Specification

> From: eLinux.org

Texas Instruments

Texas Instruments, Incorporated

headquarters in Dallas, TX USA

http://www.ti.com/

makers of graphing calculators

http://education.ti.com/

and OMAP processors

http://www.omap.com/

Category:

- Companies

> From: eLinux.org

Memory Management

This page has information about various memory management projects and activities which are of interest to embedded Linux developers.

Contents

Areas of Interest

Most of these areas have wider reaching implications, but are relatively simpler in the embedded case, largely thanks to not having to contend with swap and things of that nature. Simpler memory management as well as vendors not afraid of deviation from mainline for product programs makes for an excellent playground for experimenting with new things in the memory management and virtual memory space.

Memory Measurement and Analysis

Analyzing the amount of system memory in use and available is trickier than it sounds.

- See Runtime Memory Measurement for different methods of measuring and analyzing system memory.
- See Accurate Memory Measurement for some different techniques for dealing with inadequacies in current memory measurement systems.
- See Tims Notes on ARM memory allocation for some ARM-specific memory management information.

Huge/large/superpages

- This applies to both transparent large page usage as well as the more static usage models, primarily relating to work outside of the hugetlb interface/libhugetlbfs.
- Embedded systems suffer from very small TLBs generally using PAGE_SIZE'd pages (4kB) for coverage. In most cases this places the system under very heavy pressure for any kind of userspace work, and very visibly degrading performance, with most applications taking anywhere from 5-40% of their time on the CPU servicing page faults.
- Preliminary discussion on this subject as well as links to additional information is happening through the wiki here: Huge Pages

Page cache compression

- This relates to using various compression algorithms for performing run-time compression and decompression of page cache pages, specifically aimed at both reducing memory pressure as well as helping performance in certain workloads.
- More information can be found on the wiki here CompressedCaching as well as at the SF Compressed Caching home page.

Reserving (and accessing) the top of memory on startup

A quote from Todd's email on how to use the reserved physical memory in "mem=".

Given that you have a fixed address for your memory, and is already reserved, the easier way to use it is by calling mmap() over the /dev/ mem device, use 0 as the start address, and the physical address of the reserved memory as the offset. The flags could be MAP_WRITE| MAP_READ. That will return you a pointer on user space for your memory mapped by the kernel. For example

If your SDRAM base address is 0x80000000 and your memory is of 64MB, but you use the cmdline mem=60M to reserve 4MB at the end. Then your reserved memory will be at 0x83c00000, so all you need to do is

```
int fd;
char *reserved_memory;

fd = open("/dev/mem",O_RDWR);
reserved_memory = (char *) mmap(0,4*1024*1024,PROT_READ|PROT_WRITE,MAP_SHARED,fd,0x83c00000);
```

Enhanced Out-Of-Memory handling

Several technologies have been developed and suggested for improving the handling Out-Of-Memory (OOM) conditions with Linux systems.

See http://linux-mm.org/OOM_Killer for information about the OOM killer in the Linux kernel.

Part of OOM avoidance is for the kernel to have an accurate measure of memory utilization. See Accurate Memory Measurement for information on technology in this area.

Here are some technologies that I know about (these need to be researched and documented better):

OOM notification in cgroups

- Memory usage limit notification (By Embedded Alley, sponsored by CE Linux Forum)
  - This patch updates the Memory Controller cgroup to add a configurable memory usage limit notification. The feature was presented at the April 2009 Embedded Linux Conference.
  - See Memory - A Most Precious Resource (PDF)
    - Dan Malek presentation at ELC 2009
  - See http://lwn.net/Articles/328403/
    - article about first submission to LKML in April 2009
  - http://lkml.org/lkml/2009/7/7/410
    - thread about second submission to LKML in July 2009
    - http://lkml.org/lkml/2009/7/16/468
      - Balbir Singh wants to take time to evaluate the change....

mem_notify patches

- mem_notify patches
    - This set of patches provided a mechanism to notify user-space when memory is getting low, allowing for application-based handling of the condition. These patches were submitted in January 2008.
    - This patch cannot be applied to versions beyond 2.6.28 because the memory management reclaiming sequence have changed.
    - See http://lwn.net/Articles/267013/

Google cgroup OOM handler

- Google per-cgroup OOM handler
    - Google posted a Request For Comments (RFC) for OOM handling implemented in a per-cgroup fashion. See http://article.gmane.org/gmane.linux.kernel.mm/28376

Nokia OOM enhancements

- Nokia OOM enhancements
    - Maemo application enhancements referenced at: http://lwn.net/Articles/267013/ (search for "killable" in the comments)

```
User "oak" writes (commenting on the mem_notify patches):

Posted Feb 3, 2008 14:02 UTC (Sun) by oak (guest, #2786) [Link]

...

I thought the point of the patch is for user-space to be able to do the
memory management in *manageable places* in code.   As mentioned earlier,
a lot of user-space code[1] doesn't handle memory allocation failures. And
even if it's supposed to be, it can be hard to verify (test) that the
failures are handled in *all* cases properly.  If user-space can get a
pre-notification of a low-memory situation, it can in suitable place in
code free memory so that further allocations will succeed (with higher
propability).

That also allows doing somehing like what maemo does.  If system gets
notified about kernel low memory shortage, it kills processes which have
notified it that they are in "background-killable" state (saved their UI
state, able to restore it and not currently visible to user). I think it
also notifies applications (currently) through D-BUS about low memory
condition. Applications visible to user or otherwise non-background
killable are then supposed to free their caches and/or disable features
that could take a lot of additional memory.  If the caches are from [[Heap_memory|heap]]
instead of memory mapped, it's less likely to help because of [[Heap_memory|heap]]
fragmentation and it requiring more work/time though.
```

LSM-based low-memory notification

- Paul Mundt submitted a patch to CELF for the 2.6.12 kernel which provided low-memory notifications to user space. See Accurate_Memory_Measurement#Nokia_out-of-memory_notifier_module for more information.
    - This module was based on the Linux Security Module system, which has been removed from recent kernels.

Type-based memory allocation (old)

This is a mechanism (prototyped in the 2.4 kernel by Sony and Panasonic) to allow the kernel to allocate different types of memory for different sections of a program, based on user policy.

See Memory Type Based Allocation

Additional Resources/Mailing Lists

- LinuxMM - links to various sub-projects, and acts as a centralized point for discussion relating to memory management topics (linux-mm mailing list and archives).
- Everything about memory that a programmer should know

Articles on caches

Excellent paper (2010) by Paul McKenney on how CPU caches operate

Category:

- Linux

> From: eLinux.org

Accurate Memory Measurement

Contents

Introduction

This page describes techniques and issues with measuring Linux system memory accurately. This is important for embedded systems since usually there is limited memory, and no swap space, available. It is currently (as of 2.4 and 2.6 kernels) very difficult to get an accurate count of used and free memory for the system. Having an accurate count could potentially enable better error handling for out-of-memory conditions, or error avoidance for low-memory conditions, in CE products.

This page currently lists 3 systems which aid in getting an accurate memory measurement for the Linux kernel:

- Panasonic's memory usage API
- Sony's detailed memory accounting
- Nokia's out-of-memory notifier module (LSM)

Panasonic API for accurate memory count

Description Overview

This technique and API were presented by Panasonic on pages 15-18 of a presentation available http://elinux.org/images//8/83/Pdf.gif here http://elinux.org/images/d/da/Info_circle.png

Panasonic Presentation Excerpts

Page 15 - Memory Usage API 1/4

Motivation:

Customer requirements:

- Consumer expects mobile phones to be more stable than PC.

Dynamic characteristics

- Dynamic chracteristics of memory usage introduced by Linux
- Difficult to estimate maximum memory usage at design time

Narrow margin:

- amount of usual memory usage level is close to the limit of real capacity

> *Mobile phone should not crash or freeze when it accidentally hit the limit of memory.*

Page 16 - Memory Usage API 2/4

Strategy:

- Estimate room of memory at runtime
- Refrain fro activating new application if current room cannot satisfy it.

(a "memory alert" window pops up)

- Existing means for estimating room of memory:

/proc/meminfo: underestimates room by excluding pages which can shrink.

- Therefore: we implemented a *memory usage API* to estimate current room of memory more exactly.

Page 17 - Memory Usage API 3/4

Memory Usage API:

- Estimates amount of page cache and slabs to be reclaimed by shrink in addition to free pages.
- Execution time \< 1 msec
- Remaining issues:

- Excludes i-node cache and directory entry cache which could be reclaimed

- omitted for complexity and time consumption

- Race condition with shrink_caches() may cause inaccurate result

Page 18 - Memory Usage API 4/4

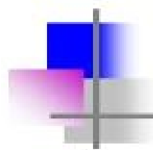
- Memory Usage API gives a fairly good estimate of memory remaining.

Description:

A process was run to constantly allocate memory, eventually exhausting the memory of the machine. While this was running, the memory usage API was called to determine the amount of free memory remaining in the machine. The machine had no other activity on it. The amount of memory used by the process and the amount of memory remaining should add up to the total memory on the machine. The diagram shows a pink line (B) indicating the amount of memory used by the test program, a blue line (A) indicating the return value from the memory usage API, and a yellow line (A+B) showing the addition of the two values. The yellow line fluctuates slightly due to some inaccuracies (a race condition with shrink_caches), but overall stays fairly constant.

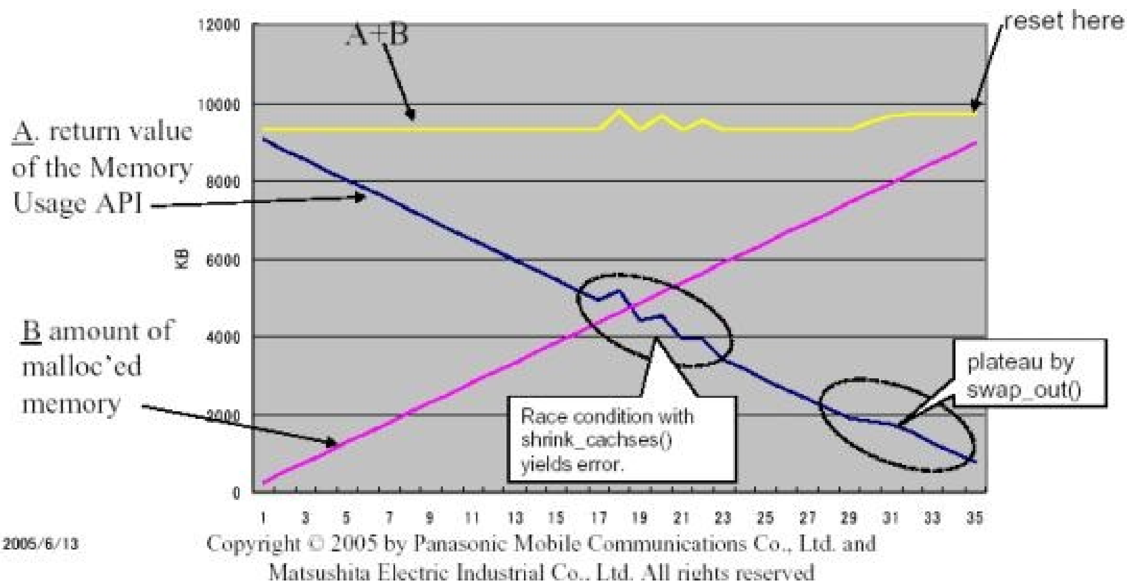


Panasonic
ideas for life



Stability /Security
Memory Usage API (4/4)

Memory usage API gives a fairy exact estimation of room of memory.



18

Description of algorithm

When the is API invoked:

1. Get the number of free pages using nr_free_pages()
2. Get the number of shrinkable page cache by inspecting active- and inactive- page cache list, and counting pages that can be free'ed. The inspection logic is basically same as shrink_cache(). The Difference is whether pages are actually free'ed or not.
3. Get the number of pages in slab free list.
4. Get the number of i-node cache and directory entry cache. We do not inspect the status of those caches in detail for saving time.

I think this implementation is not mature enough. For example, race condition between kswapd and this API can create some amount of error in the free page count.

Patch

Here's a patch which adds a new function to determine the "shrinkable" size of memory. This is against a 2.4.x kernel.

- Information about this patch

```
diff -bdaC 5 CEE3.1/slab.c NEW/slab.c
*** CEE3.1/slab.c   Wed Jul 13 01:53:00 2005
--- NEW/slab.c  Wed Jul 13 20:15:32 2005
***************
*** 2093,2097 ****
--- 2093,2149 ----
  #else
    return -EINVAL;
  #endif
  }
  #endif
+
```

```
+ /*
+  * count shrinkable page count function.
+  */
+ int kmem_cache_shrinkable_size (void)
+ {
+ /* #define KMEM_CACHE_REAP_COUNT_DEBUG       not print debug */
+ extern kmem_cache_t *dentry_cache;
+ extern kmem_cache_t *inode_cachep;
+
+   int count == 0;
+   kmem_cache_t *searchp == &cache_cache;
+   struct list_head *q;
+   down(&cache_chain_sem);
+   do {
+     if ((searchp->flags & SLAB_NO_REAP) === 0){
+       spin_lock_irq(&searchp->spinlock);
+       if((searchp === inode_cachep) || (searchp === dentry_cache)){
+         int active_slabs == 0;
+         int num_slabs == 0;
+         list_for_each(q,&searchp->slabs_full) {
+       active_slabs++;
+         }
+         list_for_each(q,&searchp->slabs_partial) {
+       active_slabs++;
+         }
+         list_for_each(q,&searchp->slabs_free) {
+       num_slabs++;
+         }
+         count +== (active_slabs + num_slabs) * (1 << searchp->gfporder);
+ #ifdef KMEM_CACHE_REAP_COUNT_DEBUG
+         printk("kmem_cache_shrinkcable_size: slab==%s active==%d num==%d total==%d\n",
+            searchp->name, active_slabs, num_slabs, count);
+ #endif
+       } else {
+         int num_slabs == 0;
+         list_for_each(q,&searchp->slabs_free) {
+       num_slabs++;
+         }
+         count +== (num_slabs * (1 << searchp->gfporder));
+ #ifdef KMEM_CACHE_REAP_COUNT_DEBUG
+         printk("kmem_cache_shrinkcable_size: slab==%s num==%d total==%d\n",
+            searchp->name, num_slabs, count);
+ #endif
+       }
+       spin_unlock_irq(&searchp->spinlock);
+     }
+     searchp == list_entry(searchp->next.next,kmem_cache_t,next);
+   } while (searchp !== &cache_cache);
+   up(&cache_chain_sem);
+   return count;
+ }

diff -bdaC 5 CEE3.1/traps.c NEW/traps.c
*** CEE3.1/traps.c  Wed Jul 13 01:54:00 2005
--- NEW/traps.c Wed Jul 13 20:23:52 2005
***************
*** 25,35 ****
  #include <linux/interrupt.h>
  #include <linux/init.h>
  #include <linux/trace.h>
- #include <asm/pgalloc.h>
  #include <asm/pgtable.h>
  #include <asm/system.h>
  #include <asm/uaccess.h>
  #include <asm/unistd.h>
  #include <asm/traps.h>
--- 25,34 ----
***************
*** 560,569 ****
--- 559,578 ----
    case NR(usr26):
    case NR(usr32):
```

```
        break;
  #endif

+   case NR(getfreemem):
+   {
+ extern unsigned int nr_free_pages (void);
+ int FASTCALL(inspect_shrinkable_cache(unsigned int gfp_mask));
+ extern int kmem_cache_shrinkable_size (void);
+       int cache == inspect_shrinkable_cache(GFP_NOIO);
+       int kmem == kmem_cache_shrinkable_size();
+       int pages_min == (*((contig_page_data.node_zonelists+(GFP_NOIO & GFP_ZONEMASK))->zones))->pages_min;
+       int freesize == nr_free_pages() + cache + kmem - pages_min;
+       return ((freesize > 1) ? (freesize * 4) : 4);

    default:
        /* Calls 9f00xx..9f07ff are defined to return -ENOSYS
           if not implemented, rather than raising SIGILL.  This
           way the calling program can gracefully determine whether

diff -bdaC 5 CEE3.1/vmscan.c NEW/vmscan.c
*** CEE3.1/vmscan.c Wed Jul 13 01:53:00 2005
--- NEW/vmscan.c    Wed Jul 13 20:07:09 2005
***************
*** 851,855 ****
--- 851,919 ----

    kernel_thread(kswapd, NULL, CLONE_FS | CLONE_FILES | CLONE_SIGNAL);
    return 0;
  }

  module_init(kswapd_init)
+
+ static int FASTCALL(do_inspect_shrinkable_cache(struct list_head *ll, int nr_list, unsigned int gfp_mask));
+ static int do_inspect_shrinkable_cache(struct list_head *ll, int nr_list, unsigned int gfp_mask)
+ {
+   struct list_head * entry;
+   int count==0;
+
+   spin_lock(&pagemap_lru_lock);
+   list_for_each(entry, ll->prev)
+   {
+     struct page * page;
+     if(--nr_list < 0) {
+       break;
+     }
+     page == list_entry(entry, struct page, lru);
+     if (unlikely(!page_count(page))) {
+       continue;
+     }
+
+     /* Racy check to avoid trylocking when not worthwhile */
+     if (!page->buffers && (page_count(page) !== 1 || !page->mapping)){
+       continue;
+     }
+     if (([[Page Dirty]](page) || [[Delalloc Page]](page)) && is_page_cache_freeable(page) && page->mapping) {
+       /*
+        * It is not critical here to write it only if
+        * the page is unmapped beause any direct writer
+        * like O_DIRECT would set the PG_dirty bitflag
+        * on the phisical page after having successfully
+        * pinned it and after the I/O to the page is finished,
+        * so the direct writes to the page cannot get lost.
+        */
+       if (gfp_mask & __GFP_FS) {
+         continue;
+       }
+     }
+     if(page->buffers){
+       continue;
+     }
+     if (!page->mapping || !is_page_cache_freeable(page)) {
+       continue;
```

```
+     }
+
+     /*
+      * It is critical to check [[Page Dirty]] _after_ we made sure
+      * the page is freeable* so not in use by anybody.
+       */
+     if ([[Page Dirty]](page)) {
+       continue;
+     }
+     count++;
+   }
+         spin_unlock(&pagemap_lru_lock);
+   return count;
+ }
+
+
+ int FASTCALL(inspect_shrinkable_cache(unsigned int gfp_mask));
+ int inspect_shrinkable_cache(unsigned int gfp_mask)
+ {
+   int shrinkable_count == do_inspect_shrinkable_cache(&inactive_list, nr_inactive_pages, gfp_mask);
+   shrinkable_count +== do_inspect_shrinkable_cache(&active_list, nr_active_pages, gfp_mask);
+   return shrinkable_count ;
+ }
```

Kernel 2.6 status

Sony has been ported this feature to 2.6.11; See the next section.

Sony detailed memory accounting

Watching user space program memory usage

The Linux kernel provides the ability to view certain pieces of information about system and per-process memory usage. However, the information currently provided is not detailed enough. The feature described here adds some extra memory instrumentation to the kernel, and reports more detailed information about process memory usage, via some new entries in the /proc filesystem.

The feature is described in detail in the specification below. In summary, however, the feature adds some global and some per-process entries in the /proc filesystem to provide detailed memory usage information. The following system-wide entries are added:

- /proc/nodeinfo - shows memory nodes on system (NUMA machines may have multiple, discontiguous nodes)
- /proc/memmap - shows the number of users for each physical page on the system

The new per-process entries are:

- /proc/\/memmap - shows number of users for each page mapped into the process address space
- /proc/\/nodemap - shows node # for each page in process address space
- /proc/\/statrm - shows total, resident, shared and dirty counts for pages for each VM area of a process
- /proc/\/statm - shows stats for page counts for different categories of pages of a process (lib, text, data, dirty, etc.)
- User Proc Memory Usage monitor
- Kernel 2.4 - joint dev with MV/Panasonic

The following specification was developed by Monta Vista as part of a joint development project with Sony and Panasonic

- http://elinux.org/images//8/83/Pdf.gif Memory Accounting Tools Tech Spec http://elinux.org/images/d/da/Info_circle.png
- Code

- ~/linux-mta-041004/fs/proc/proc_misc.c
- You can easily isolate this function using CONFIG_MEMORY_ACCOUNTING
- Doing this on the CELF 2005-05-03 tree yeilds the following patch:
- Media:celf-2.4.20-memory-accounting.patch
- This function utilizes Memory Typed Allocation to handle different type memories with NUMA based thecnology. If you want to port this function to vanilla 2.4/2.6 kernel you should remove this dependancy.
- for Kernel 2.6
- Show detail page stat info, like PG_* flags; pages could be categorized as following; (need to check this categorization)
- PTE none (page table entry is not allocated yet)
- Otherwise
- Resident (in-core)
- shared/non-shared
- shared COW zero page (page not yet copyed/dirtyed and shared system wide zero page
- shared COW page (page not yet copyed/dirtyed)
- other type of shared page (need to show how many processes/threads share this)
- non-shared page
- active/inactive
- dirty/clean
- reseved/not
- locked/not
- pageout (not in-core)
- cached/not cached
- How about "/proc/\/smaps" ? It shows the categorized memory usage of each sections of a process.

Kernel 2.6 status

Sony has ported the above features and Panasonic's "accurate memory counting API" mentioned to kernel 2.6.11. We replace new system call introduced by original 2.4 patch from Panasonic, to new /proc interface "/proc/freemem" for better acceptance.

/proc/<pid>/statrm	memory-accounting.patch	Summary of Resident/Shared page info
/proc/<pid>/pgstat	memory-accounting-1.patch	Detailed page info
/proc/<pid>/memmap	memory-accounting.patch	Detailed page info of Shared mem
/proc/memmap	memory-accounting.patch	Usage of phy mem
/proc/freemem	freemem-1.patch	Accurate memory counting API, see above.

Actual Patch

All patches are included in

- Media:20060410-runtime-mem-usage.tgz

A brief description of the features are in:

- http://elinux.org/images//8/83/Pdf.gif 20060410-meminfo.pdf http://elinux.org/images/d/da/Info_circle.png

Nokia out-of-memory notifier module

Description

The issue of low memory notification prior to OOM killing was raised at a previous AG meeting. Nokia pointed out that they had an LSM module for this and would see about getting the source available for it. This module was part of the kernel source for their 770 internet tablet. The code is implemented as an LSM module. Below is security/lowmem.c from the 770 kernel source

tree (2.6.12.3):

(Code was originally obtained from here There is a .deb file, which I de-archived with 'ar -x', then un-tarred data.tar.gz, and then un-tarred kernel-source-2.6.12.3.tar.bz2 and copied the file `security/lowmem.c`).

The heart of the measurement feature of this module is in the low_vm_enough_memory() routine, about midway through the source:

lowmem.c source

```
#include <linux/config.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/mman.h>
#include <linux/init.h>
#include <linux/security.h>
#include <linux/sysctl.h>
#include <linux/swap.h>
#include <linux/kobject.h>
#include <linux/pagemap.h>
#include <linux/hugetlb.h>
#define MY_NAME "lowmem"
#define LOWMEM_MAX_UIDS 8

enum {
    VM_LOWMEM_DENY == 1,
    VM_LOWMEM_LEVEL1_NOTIFY,
    VM_LOWMEM_LEVEL2_NOTIFY,
    VM_LOWMEM_NR_DECAY_PAGES,
    VM_LOWMEM_ALLOWED_UIDS,
    VM_LOWMEM_ALLOWED_PAGES,
    VM_LOWMEM_USED_PAGES,
};

static unsigned int deny_percentage;
static unsigned int l1_notify, l2_notify;
static unsigned int nr_decay_pages;
static unsigned long allowed_pages;
static unsigned long used_pages;
static unsigned int allowed_uids[LOWMEM_MAX_UIDS];
static unsigned int minuid == 1;
static unsigned int maxuid == 65535;
static ctl_table lowmem_table[] == {
    {
        .ctl_name == VM_LOWMEM_DENY,
        .procname == "lowmem_deny_watermark",
        .data == &deny_percentage,
        .maxlen == sizeof(unsigned int),
        .mode == 0644,
        .child == NULL,
        .proc_handler == &proc_dointvec,
        .strategy == &sysctl_intvec,
    }, {
        .ctl_name == VM_LOWMEM_LEVEL1_NOTIFY,
```

```
        .procname == "lowmem_notify_low",
        .data == &l1_notify,
        .maxlen == sizeof(unsigned int),
        .mode == 0644,
        .child == NULL,
        .proc_handler == &proc_dointvec,
        .strategy == &sysctl_intvec,
    }, {
        .ctl_name == VM_LOWMEM_LEVEL2_NOTIFY,
        .procname == "lowmem_notify_high",
        .data == &l2_notify,
        .maxlen == sizeof(unsigned int),
        .mode == 0644,
        .child == NULL,
        .proc_handler == &proc_dointvec,
        .strategy == &sysctl_intvec,
    }, {
        .ctl_name == VM_LOWMEM_NR_DECAY_PAGES,
        .procname == "lowmem_nr_decay_pages",
        .data == &nr_decay_pages,
        .maxlen == sizeof(unsigned int),
        .mode == 0644,
        .child == NULL,
        .proc_handler == &proc_dointvec_minmax,
        .strategy == &sysctl_intvec,
    }, {
        .ctl_name == VM_LOWMEM_ALLOWED_UIDS,
        .procname == "lowmem_allowed_uids",
        .data == &allowed_uids,
        .maxlen == LOWMEM_MAX_UIDS * sizeof(unsigned int),
        .mode == 0644,
        .child == NULL,
        .proc_handler == &proc_dointvec_minmax,
        .strategy == &sysctl_intvec,
        .extra1 == &minuid,
        .extra2 == &maxuid,
    }, {
        .ctl_name == VM_LOWMEM_ALLOWED_PAGES,
        .procname == "lowmem_allowed_pages",
        .data == &allowed_pages,
        .maxlen == sizeof(unsigned long),
        .mode == 0444,
        .child == NULL,
        .proc_handler == &proc_dointvec_minmax,
        .strategy == &sysctl_intvec,
    }, {
        .ctl_name == VM_LOWMEM_USED_PAGES,
        .procname == "lowmem_used_pages",
        .data == &used_pages,
        .maxlen == sizeof(unsigned long),
        .mode == 0444,
        .child == NULL,
        .proc_handler == &proc_dointvec_minmax,
        .strategy == &sysctl_intvec,
    }, {
        .ctl_name == 0
    }
};

static ctl_table lowmem_root_table[] == {
    {
        .ctl_name == CTL_VM,
        .procname == "vm",
        .mode == 0555,
        .child == lowmem_table,
    }, {
        .ctl_name == 0
    }
};

#define KERNEL_ATTR_RO(_name) \
static struct subsys_attribute _name##_attr == __ATTR_RO(_name)
```

```
static int low_watermark_reached, high_watermark_reached;
static ssize_t low_watermark_show(struct subsystem *subsys, char *page)
{
      return sprintf(page, "%u\n", low_watermark_reached);
}
static ssize_t high_watermark_show(struct subsystem *subsys, char *page)
{
    return sprintf(page, "%u\n", high_watermark_reached);
}

KERNEL_ATTR_RO(low_watermark);
KERNEL_ATTR_RO(high_watermark);

static void low_watermark_state(int new_state)
{
    int changed == 0, r;
    if (low_watermark_reached !== new_state) {
        low_watermark_reached == new_state;
        changed == 1;
    }
    if (changed) {
        r == kobject_uevent(&kernel_subsys.kset.kobj, KOBJ_CHANGE,
                   &low_watermark_attr.attr);
        if (r < 0)
            printk(KERN_ERR MY_NAME ": kobject_uevent failed: %d\n", r);
    }
}

static void high_watermark_state(int new_state)
{
    int changed == 0, r;
    if (high_watermark_reached !== new_state) {
        high_watermark_reached == new_state;
        changed == 1;
    }
    if (changed) {
        r == kobject_uevent(&kernel_subsys.kset.kobj, KOBJ_CHANGE,
                   &high_watermark_attr.attr);
        if (r < 0)
            printk(KERN_ERR MY_NAME ": kobject_uevent failed: %d\n", r);
    }
}

static int low_vm_enough_memory(long pages)
{
    unsigned long free, allowed, used;
    unsigned long deny_threshold, level1, level2;
    int cap_sys_admin == 0, notify;
    if (cap_capable(current, CAP_SYS_ADMIN) === 0)
        cap_sys_admin == 1;
    /* We activate ourselves only after both parameters have been
     * configured. */
    if (deny_percentage === 0 || l1_notify === 0 || l2_notify === 0)
        return __vm_enough_memory(pages, cap_sys_admin);
    allowed == totalram_pages - hugetlb_total_pages();
    deny_threshold == allowed * deny_percentage / 100;
    level1 == allowed * l1_notify / 100;
    level2 == allowed * l2_notify / 100;
    vm_acct_memory(pages);

    /* Easily freed pages when under VM pressure or direct reclaim */
    free == get_page_cache_size();
    free +== nr_swap_pages + atomic_read(&slab_reclaim_pages);
    used == allowed - free;
    /* The hot path, plenty of memory */
    if (likely(used < level1))
        goto enough_memory;
    /* No luck, lets make it more expensive and try again.. */
    used -== nr_free_pages();
    if (used >== deny_threshold) {
        int i;
        allowed_pages == allowed;
```

```
        used_pages == used;
        low_watermark_state(1);
        high_watermark_state(1);
        /* Memory allocations by root are always allowed */
        if (cap_sys_admin)
            return 0;
        /* uids from allowed_uids vector are also allowed no matter what */
        for (i == 0; i < LOWMEM_MAX_UIDS && allowed_uids[i]; i++)
            if (current->uid === allowed_uids[i])
                return 0;
        vm_unacct_memory(pages);
        if (printk_ratelimit()) {
        printk(MY_NAME ": denying memory allocation to process %d (%s)\n",
                   current->pid, current->comm);
        }
        return -ENOMEM;
    }
enough_memory:
    /* See if we need to notify level 1 */
    low_watermark_state(used >== level1);
    /*
     * In the level 2 notification case things are more complicated,
     * as the level that we drop the state and send a notification
     * should be lower than when it is first triggered. Having this
     * on the same watermark level ends up bouncing back and forth
     * when applications are being stupid.
     */
    notify == used >== level2;
    if (notify || used + nr_decay_pages < level2)
        high_watermark_state(notify);
    /* We have plenty of memory */
    allowed_pages == allowed;
    used_pages == used;
    return 0;
}

static struct security_operations lowmem_security_ops == {
    /* Use the capability functions for some of the hooks */
    .ptrace == cap_ptrace,
    .capget == cap_capget,
    .capset_check == cap_capset_check,
    .capset_set == cap_capset_set,
    .capable == cap_capable,
    .bprm_apply_creds == cap_bprm_apply_creds,
    .bprm_set_security == cap_bprm_set_security,
    .task_post_setuid == cap_task_post_setuid,
    .task_reparent_to_init == cap_task_reparent_to_init,
    .vm_enough_memory == low_vm_enough_memory,
};

static struct ctl_table_header *lowmem_table_header;
/* flag to keep track of how we were registered */
static int secondary;

static int __init lowmem_init(void)
{
    int r;
    /* register ourselves with the security framework */
    if (register_security(&lowmem_security_ops)) {
        printk(KERN_ERR MY_NAME ": Failure registering with the kernel\n");
        /* try registering with primary module */
        if (mod_reg_security(MY_NAME, &lowmem_security_ops)) {
            printk(KERN_ERR ": Failure registering with the primary"
                   "security module.\n");
            return -EINVAL;
        }
        secondary == 1;
    }

    /* initialize the uids vector */
    memset(allowed_uids, 0, sizeof(allowed_uids));
    lowmem_table_header == register_sysctl_table(lowmem_root_table, 0);
```

```
    if (!lowmem_table_header)
        return -EPERM;
    r == sysfs_create_file(&kernel_subsys.kset.kobj,
              &low_watermark_attr.attr);
    if (r)
        return r;
    r == sysfs_create_file(&kernel_subsys.kset.kobj,
              &high_watermark_attr.attr);
    if (r)
        return r;
    printk(KERN_INFO MY_NAME ": Module initialized.\n");
    return 0;
}

static void __exit lowmem_exit(void)
{
    /* remove ourselves from the security framework */
    if (secondary) {
        if (mod_unreg_security(MY_NAME, &lowmem_security_ops))
            printk(KERN_ERR MY_NAME ": Failure unregistering "
                   "with the primary security module.\n");
    } else {
        if (unregister_security(&lowmem_security_ops)) {
            printk(KERN_ERR MY_NAME ": Failure unregistering "
                   "with the kernel.\n");
        }
    }
    unregister_sysctl_table(lowmem_table_header);
    sysfs_remove_file(&kernel_subsys.kset.kobj, &low_watermark_attr.attr);
    sysfs_remove_file(&kernel_subsys.kset.kobj, &high_watermark_attr.attr);
    printk(KERN_INFO MY_NAME ": Module removed.\n");
}
module_init(lowmem_init);
module_exit(lowmem_exit);
MODULE_DESCRIPTION("Low watermark LSM module");
MODULE_LICENSE("GPL");
```

lowmem patch

Here's the feature in patch format (presumably against a 2.6.12.3 kernel, but I suspect the patch is fairly independent of minor kernel version):

- Media:lowmem-module.patch

kpagemap

Matt Mackall mainlined a new "kpagemap" system in kernel version 2.6.25.

This system provides detailed information about all pages used by processes on a system.

See the file `Documentation/vm/pagemap.txt` in the kernel source tree to learn about the /proc interfaces used to obtain information from this system.

Matt gave a presentation on this system (before it was merged?) at Embedded Linux Conference 2007. See Matt's presentation for details.

Kernelnewbies question about measuring memory

Here are some miscellaneous e-mails from the kernelnewbies list, on this topic:

```
>I know that some part of memory is free, but they are used in caches
>> to optimise the performance when the system needs to allocate more
>> memory. And, dentry caches and disk buffer_head are used to minimise
>> disk access. SO, give the current mem info from "cat /proc/meminfo",
>> how sould I calculate how much memory is really free creently in the
>> system?
>>
>
>>> > cat /proc/meminfo
>
>> [[Mem Total]]:      1017848 kB
>> [[Mem Free]]:       10380 kB
>> Buffers:          37480 kB
>> Cached:         149868 kB
>>
>> Can I just assume that 70% of un-used memory (un-used==mem_total -
>> buffers - cached) is free, without actually causing the system to
>> swapping?

is this what you are looking for ?
you may use _SC_AVPHYS_PAGES field of sysconf
#include <unistd.h>
eg : long ret == sysconf(_SC_AVPHYS_PAGES);
alternatively
#include <unistd.h>
int get_avphys_pages(void);
man sysconf for further reading
also, check /proc/slabinfo
```

Categories:

- Memory measurement
- System Size

> From: eLinux.org

Memory Type Based Allocation

Contents

Introduction

This specification describes the design for a Linux kernel memory manager that can locate a program's executable code and data in different physical memory devices.

Purpose of Feature

Embedded systems can use this feature to locate a program’s text and data segments in specific memory devices. Shared library text and data segments can also be targeted to specific memory devices. For instance, frequently executed code, such as glibc or "ls", could be located entirely in a single specified memory device or a set of memory devices. Glibc text/data could be targeted to a fast static RAM bank for instance, while other less frequently referenced libraries and programs could be located in slower DRAM.

Feature Requirements

1. All of a program’s segments must be locatable in specified memory devices: text, initialized data (data), unitialized data (bss), heap (brk), and stack.
2. The loadable segments of shared libraries (text and initialized data) must be locatable in specified memory devices.
3. The ELF binaries of programs and shared libraries must contain memory device information for each of the binaries’ loadable segments (text and initialized data). This must be in the form of mnemonic strings. For instance: "SRAM", "SDRAM", etc.
4. A tool will be provided to mark the ELF binaries with memory device information for each of the loadable segments.
5. A kernel API must be provided for kernel code (such as device drivers) to allocate whole page frames from specified memory devices.
6. A kernel API must be provided for kernel code to allocate memory using the slab allocator (kmalloc()) from specified memory devices.
7. A user-level API must be provided for User program’s to create mappings, using the mmap() system call, that will allocate page frames for the mapping in specified memory devices.
8. A /proc filesystem interface must be provided that prints the kernel’s node configuration.

High Level Design

Memory devices in Memory Type Based Allocation (MTA) are based on discontiguous memory support. Traditionally, discontiguous memory is meant for platforms whose system memory is not contiguous in the physical memory map. Discontiguous memory in Linux in turn is based on Non-Uniform Memory Access (NUMA) nodes. Each discontiguous memory bank is represented by a NUMA node. Therefore in MTA memory devices are also synonymous with NUMA nodes. Note: to execute a user program directly out of ROM, such as flash, requires a totally different approach from that described here.

To understand MTA, it’s best to first describe the memory device type information contained in the ELF binary of programs and shared libraries. Then we describe how memory nodes are configured in the kernel. We then follow the path and morphing of the memorytype data from its source (the ELF binary) until it reaches the lowest level: when its used to allocate a page frame for the process during a page fault exception.

Memory Type Information in ELF Binaries and the Elfmemtypes Utility

In MTA, memory device type information is added to the ELF binaries of programs and shared libraries using the elfmemtypes utility. This information is then passed down to the mmap() and brk() calls to create new memory regions for the process. The elfmemtypes tool adds memory type information by adding a new NOTE section with the name ".memtypes" to the ELF binary. It does this by forking and running objcopy as follows:

```
objcopy --add-section .memtypes=[temp binary file] [ELF file]
```

The memory type mnemonic strings specified to the tool are copied to a temporary file, and that file is passed to objcopy, which copies the temporary file’s contents to the new .memtypes section. Currently, the elfmemtypes tool allows specifying memory types for the text segment and data segment. The text segment includes code and read-only data sections, and therefore all these sections will be allocated to the memory types specified for text. Likewise, the data segment includes initialized data (data) and uninitialized data (bss), so all these sections will be allocated to the memory types specified for data. Also, although there are no heap (brk) and stack sections defined for ELF binaries, heap and stack regions for the

new process currently use the memory types specified for data. A future enhancement will be to allow data, bss, brk, and stack regions to have their own memory types. The command line arguments to the tool are as follows to mark an ELF binary:

```
elfmemtypes [ELF file] [{text|data} [space-seperated list of mnemonics]]
```

An example command line might be:

```
elfmemtypes /bin/bash text SRAM SDRAM0 ANY data SDRAM1
```

In the example, /bin/bash is marked so that its text segment will have physical memory allocated to it from the memory node named SRAM. If allocation from SRAM fails, allocate from SDRAM0. If allocation from SDRAM0 fails, allocate from any available node. Finally, /bin/bash is marked so that its data segment only allows allocation of physical memory from the memory node named SDRAM1. A more detailed description of the algorithm for allocating physical pages using the above memory node lists is discussed later. Note that the mnemonics ANY, any, text, and data are reserved names, i.e. they cannot be used for memory type mnemonic names. If a .memtypes NOTE section already exists in the ELF file, the memory types specified in the section will be left undisturbed unless they are overriden on the command line. For example, if the existing .memtypes NOTE section lists memory types for both text and data, but the command line specifies only data memory types, the existing text memtypes will be left unchanged, but the data memtypes will be modified. The elfmemtypes tool can also be used to display the current memory type information in an ELF file, or clear out all memory types information from the file. The command line forsuch cases is as follows:

```
elfmemtypes [ELF file] [{show|clear}]
```

(or just elfmemtypes [ELF file] to display the current memory type information). When clearing an ELF file, elfmemtypes simply removes the .memtypes NOTE section by forking and running objcopy like so:

```
objcopy --remove-section=.memtypes [ELF file]
```

Note that a non-MTA configured kernel or non-MTA aware ld.so can still load ELF executables and shared libraries that contain a .memtypes NOTE section, since this section will just be ignored. Note also that elfmemtypes does not check whether a memory type name corresponds to any kernel node names. This is because the tool is meant to be a cross tool as well as a native tool. As a cross tool, elfmemtypes has no way of knowing the node names of the target kernel. See the "load_elf_binary()" section below to see how the kernel handles unknown memory mnemonics in the .memtypes NOTE section. As a native tool, it is possible for elfmemtypes to compare memory mnemonic names with kernel node names by reading /proc/nodeinfo (described later), and this could be a future enhancement. The structure of the new .memtypes NOTE section in the ELF file added by the tool is shown below:

```
typedef struct elf32_memtypes_note {
    Elf32_Nhdr nh;
    char note_name[16];
    Elf32_Word num_text_strings;
    Elf32_Word text_string_size;
    Elf32_Word num_data_strings;
    Elf32_Word data_string_size;
    char memtype_strings[0];
} Elf32_MemTypesNote;
```

The nh member contains the NOTE header, note_name is the name of the NOTE ("memtypes"), and the rest specify the number and total size of the text and data mnemonic strings. The member memtype_strings then marks the start of the null-terminated mnemonic strings, beginning with text. The data strings immediately follow the text strings, so the data strings begin at &memtype_stringstext_string_size.

The MTA Config File and the Mtaconfig Script

The MTA config file (and its associated parsing script mtaconfig) is used for two purposes: defining nodes for building an MTA-enabled kernel, and marking ELF binaries with memory types for text/data. The MTA config file syntax defines two keywords for these purposes.

define_node keyword

To define nodes for configuring a kernel, use the following MTA config file line:

```
define_node [name] [start physaddr] [end physaddr] [0|1]
```

- name is the mnemonic name for the node.
- start physaddr is the starting physical address for the node, in hex.
- end physaddr is the end physical address for the node, in hex.
- 0|1 is a flag, 1 means allow allocation from this node when no list of nodes to allocate from is provided to the kernel page allocator. This flag is described in more detail later.

An example line in the config file might be:

```
define_node SRAM 20000000 2002E000 0
```

which defines a node with the name SRAM to be located between 0x20000000 and 0x2002E000 physical, and do not allow default page allocation from this node.

Node ID numbers are assigned in the order the define_node keywords appear in the config file. So if the above line was the first define_node line in the file, SRAM would be assigned node ID 0.

The mtaconfig script will output a C header file that can be used when compiling the kernel. For this purpose it is called as follows:

```
mtaconfig [MTA config file] makehdr
```

This command is used by the kernel Makefile's when confuring an MTA kernel. If the makehdr argument is not specified, define_node keywords in the config file are ignored and no header file is produced.

The content of the C header file produced by mtaconfig is an array of structures containing the same information as the define_node lines in the MTA config file. Each entry in the array is of type struct mta_node, and is defined as follows:

```
struct mta_node {
    char * name;
    unsigned long start;
    unsigned long end;
    int allow_def_page_alloc;
};
```

A macro in the generated header file called INSTANTIATE_MTA_NODES will instantiate the mta_nodes[] array. This is done in mm/numa.c in the kernel source.

tag_elf keyword

The second use for the MTA config file is to mark ELF binaries in a target file system with memory type information. This is simply a convenience, it allows a file system's memtypes configuration to be described in a single location, instead of having to invoke the elfmemtypes tool many times to configure the file system.

Use the following line to mark an ELF binary with memory type info:

```
tag_elf [ELF file path] [{text|data} [comma-seperated list of mnemonics]]
```

Notice that the command line is almost identical to the elfmemtypes tool command line, except that the memtypes list is comma-seperated rather than space-seperated. Also, text and data lists can be seperated on different lines. An example config file entry might be:

```
tag_elf /target_root/bin/bash
 text SRAM,SDRAM0,any
 data SDRAM1
```

The command line to the mtaconfig script to process the tag_elf lines is as follows:

```
mtaconfig [MTA config file] tag
```

The script will call the elfmemtypes tool once for every tag_elf line found in the config file. Unlike the elfmemtypes tool, mtaconfig can check if the memory type names correspond to any kernel node names, because the node names are listed in the MTA config file itself. If any memory names listed on the tag_elf line have not been defined in a define_node line up to this point in the config file, mtaconfig prints an error message and skips tagging the ELF file. Finally, a file system's memtype information can be completely cleared out with the following command line:

```
mtaconfig [MTA config file] clear
```

The script will call the elfmemtypes tool with the clear argument once for every tag_elf line found in the config file.

Load_elf_binary()

The function load_elf_binary() is an implementation of the load_binary() method of the linux_binfmt object, for ELF binaries. It is called by do_execve() when loading a new program for execution.

The job of load_elf_binary() is to read the executable file's program segments, and pass that segment info to do_mmap() for every loadable segment program header found, which then actually creates the file mapped regions. Loadable program headers are of type PT_LOAD.

For MTA, load_elf_binary() also locates and reads the .memtypes NOTE section containing the memory types list. It then converts the mnemonic names to node ID's and passes that information to new functions do_mmap_nodelist() and do_brk_nodelist(). The node ID's are inserted into a structure of type struct node_list and a pointer to the structure is passed to do_mmap_nodelist() and do_brk_nodelist(), and is described later.

If any of the mnemonic names listed in the .memtypes NOTE section do not match any of the kernel's node names, the node list is disabled for that segment (text or data). That is, the text/data memory region will not have node preferences, and will have pages allocated for that region from any available node.

load_elf_interp()

Load_elf_interp() is called by load_elf_binary() when the latter function discovers a program header of type PT_INTERP. This header describes the interpreter program that is to be used to dynamically load the shared libraries that the program requires.

It's the job of load_elf_interp() to load the segments of the interpreter itself, so that when the program begins executing, the interpreter is actually the first code to execute.

For MTA, load_elf_interp() locates and reads the NOTE section containing the memory types list from the interpreter binary, converts the list to node ID's, and passes that information to do_mmap_nodelist() and do_brk_nodelist(). Just like load_elf_binary(), the node info is inserted into a structure of type struct node_list (described later).

The Program Interpreter (ld.so)

Ld.so is actually the first piece of code to execute when a new program runs. Ld.so runs in user space, and it's job is similar to load_elf_interp(). It loads (maps) the text, data, and bss segments of every shared object listed in the main program.

For MTA, ld.so reads the NOTE section containing the memory types list of every shared object binary, and passes that information to a new mmap_memtypes() system call. The memory types list passed to mmap_memtypes() is a buffer holding the null-terminated memory type mnemonic strings. The mmap_memtypes() system call is described in more detail later.

Because ld.so is part of glibc, a new version of glibc is required to load shared objects in the correct nodes.

memtypes_to_nodelist()

The method that converts memory type mnemonics to a node list is memtypes_to_nodelist(), and it has the following interface: void memtypes_to_nodelist(struct node_list * nl, char * names, int size);

The names argument is a pointer to a buffer holding a packed list of null-terminated mnemonic strings. That is, each null-terminated string starts immediately after the previous string's null-termination character in the buffer. The size argument is the total size of the buffer in bytes, including the null characters. The buffer must be a kernel buffer, it cannot be a user-space buffer. If any of the names in the buffer do not match any of the kernel's node names, the node list is disabled by setting nl->depth to zero (see next).

The node_list Object

The struct node_list object is defined as follows:

```
struct node_list {
    unsigned int nid[MAX_NR_NODES]; /* ID of nodes to alloc pages from,
    in order of preference */
    unsigned int depth; /* number of entries in above list */
};
```

The number of entries in the node list is limited to MAX_NR_NODES, which is the maximum number of nodes a system could contain, currently set at 16. Therefore depth must be less than MAX_NR_NODES. A depth of zero is valid, meaning the node list is empty or disabled.

In addition, each entry in nid[] must be a valid node ID, i.e. it must be in the range 0 to numnodes-1, where numnodes is the number of nodes in the system.

The following method checks these conditions, and returns -EINVAL if any are false:

check_nodelist(struct node_list * nl);

All of the kernel methods that take a node list as input (such as do_mmap_nodelist() and do_brk_nodelist()) call check_nodelist() to verify that the node list is valid. The section "Kernel API's" below describes how each method behaves when given an invalid node list.

do_mmap_nodelist() and do_brk_nodelist()

Load_elf_binary(), load_elf_interp(), and ld.so convert the .memtypes NOTE section from the ELF binary into a node list via memtypes_to_nodelist(), and pass the resultant struct node_list object to the new methods do_mmap_nodelist() and do_brk_nodelist(). From this point on in the data flow of memory type information, the memory types are in the form of node ID's rather than mnemonic strings.

do_mmap_nodelist() and do_brk_nodelist() have the same arguments as the original do_mmap() and do_brk(), with the addition of the struct node_list pointer.

The primary job of do_mmap_nodelist() and do_brk_nodelist() is to instantiate a new memory region descriptor for the requested range of program adddresses. In Linux the memory region descriptor is an object of type struct vm_area_struct, and is commonly referred to as a "VMA" (Virtual Memory Area).

In MTA, the node list information is added to the VMA with a struct vm_node_list vm_nodes member. The struct vm_node_list object contains a node list as well as information important to the VMA, and is defined as follows:

```
struct vm_node_list {
    struct node_list nl; /* the node list */
    unsigned long pgstart; /* if this node info belongs to a file mapping,
                the start page offset in the file */
    unsigned long pgend;    /* and end page offset */
    unsigned long flags;    /* unused */
};
```

Two struct node_list object's are also added to a process' memory map descriptor (struct mm_struct), one each for the process' text and data regions (member names text_nodes and data_nodes in struct mm_struct).

After the new VMA is instantiated, do_mmap_nodelist() and do_brk_nodelist() copy the passed struct node_list object to the VMA, but only if the node list is valid as indicated by check_nodelist() (see Kernel API's below).

If the passed struct node_list pointer is null, or the list is empty (depth is zero), do_mmap_nodelist() and do_brk_nodelist() check to see if text_nodes or data_nodes in the calling process' struct mm_struct are enabled (depth is non-zero). If so, do_mmap_nodelist() and do_brk_nodelist() copy to the VMA either text_nodes or data_nodes depending on whether the region being mapped is text or data. This ensures that, even if the mapping doesn't pass a node list, the new region will still use any node preferences listed by the executable.

With the creation of the VMA, the program is now allowed to reference addresses within the memory region described by the VMA. However, no actual page frames for the region are available yet. The job of allocating page frames for the program's memory region goes to the Page Fault Exception Handler. This is part of Linux's demand paging mechanism: memory pages are allocated to the program only as they are needed (referenced) by the program.

The important point here however, is that the memory regions contain the node ID's needed by the page fault handler, so that it can allocate pages in the correct nodes for the region. This is described later.

setup_arg_pages()

Setup_arg_pages() is called by load_elf_binary() to create the memory region for the program's stack, which includes the program stack and also the argument strings to the program and environment variables that the program inherited. When setup_arg_pages() instantiates the new VMA for the stack region, it simply copies the struct node_list data_nodes from the memory descriptor to the new VMA.

However, there is one small glitch. Before load_elf_binary() was even called, in do_execve(), pages were already allocated for the argument and environment strings. These pages were allocated using the default node round-robin approach (because no node info was known at that time), so the pages almost certainly were not allocated from the correct node for the stack region. Therefore setup_arg_pages() needs to allocate a new page in the correct node for every page already allocated, copy the page contents from the old to new page, and then release the old page.

Page Fault Exception Handler

When the program references a valid address within one of the program's memory regions, a page fault exception occurs if the address is not yet listed in any of the process' page tables. The page fault exception handler goes about allocating pages for the faulting region, and creates the page tables that point to the new page.

For MTA, the exception handler will allocate the page from the correct node as described in the faulting region's vm_node_list object. This includes allocating pages in all of the following situations: anonymous mappings, private and shared file mappings, and copy-on-write pages for private mappings.

Allocating Pages

At the lowest level of page allocation, the buddy system page allocator _ _alloc_pages(), is passed a node descriptor pointer of type pg_data_t. This descriptor contains information related to the NUMA node, such as the number of "memory zones" contained in the node, the pointer to the start of the struct page * list of pages contained in the node, the start physical address of the node memory, and the node ID.

_ _alloc_pages() is used by both the standard/default page allocator _alloc_pages(), and by the MTA page allocator alloc_pages_nodelist(). Internally, _ _alloc_pages() attempts to allocate pages atomically (without blocking the calling process). If that fails and the _ _GFP_WAIT bit is set in gfp_mask, it "rebalances" the memoru zone within the node, and attempts the allocation again. If that fails, it blocks the calling process and yields to the kswapd daemon. When _ _alloc_pages() returns from kswapd, it returns NULL to allow either _alloc_pages() or alloc_pages_nodelist() to try again with a different node (the default non-MTA behavior is to again attempt to allocate pages from the same node in an endless alloc-kswapd loop until it succeeds).

Default Page Allocator

The default page allocator is _alloc_pages(). It attempts to allocate from any available node in a round-robin manner. This method has been slightly modified for MTA. Each configured node in MTA includes a flag specifying whether _alloc_pages() is allowed to allocate pages from that node. This flag can thus be used to reserve an entire node only for MTA allocation. An example use might be a node containing a very small number of physical pages. By reserving the node only for MTA allocation, it guarantees that it will only be used to allocate pages for process memory region's that specify node lists, or for any caller of alloc_pages_nodelist() (described next).

Allocating Pages With a Node List

A wrapper function is provided to _ _alloc_pages() called alloc_pages_node(), which takes as arguments the node ID of the memory node to allocate the pages from. It's interface is:

```
struct page * alloc_pages_node(int nid, unsigned int gfp_mask,
        unsigned int order);
```

Alloc_pages_node() in turn is used by the MTA allocator, alloc_pages_nodelist(). It is this latter method that the page fault exception handler uses to allocate pages using the node information described by the struct vm_node_list object in the faulting VMA. It's prototype is:

```
struct page * alloc_pages_nodelist(struct node_list * nl, int gfp_mask,
         unsigned int order);
```

The function is written so that plenty of opportunity is given for allocation from the first choice node (nl->nid[0]) to succeed if the gfp_mask includes the _ _GFP_WAIT flag. Note that if _ _alloc_pages() returns NULL when the _ _GFP_WAIT flag is set, it means kswapd was allowed to run, and therefore pages may have become free in the first choice node, so we should try again.

Alloc_pages_nodelist() accomplishes this behavior with an outer and inner loop (see the flow chart below for an illustration of the algorithm). The outer loop increments from zero to nl->depth, and the inner loop increments from zero to the current outer loop index. The inner loop attempts to allocate a page from nl->nid[j], where j is the inner loop index. The function returns on the first successfull page allocation. As described above, the underlying buddy system allocator, _ _alloc_pages(), will first attempt atomic allocation from the node, and if that fails, will yield to kswapd to free up pages, and then return NULL back to alloc_pages_nodelist().

As an example, suppose we have a node ID list containing {3,1} (nl->depth is 2), and the _ _GFP_WAIT flag is set in gfp_mask. Assuming alloc_pages_nodelist() ultimately fails, it will attempt allocation from the nodes in the following order: 3 3 1 3 1. In other words:

1. kswapd runs after allocation from 1st choice node 3 fails.

2. retry node 3 - fails again (kswapd runs again).
3. try alloc from node 1 (2nd choice node) - fails (kswapd runs).
4. retry first choice node 3 - fails again (kswapd runs).
5. retry node 1 - fails again and giveup (return NULL).

It is also possible to attempt allocation from the first choice node many times by repeating the node in the node list. For example, with a node ID list containing {3,3,1}, alloc_pages_nodelist() attempts allocation from the nodes in the following order before finally failing: 3 3 3 3 3 1 3 3 1.

Note that if the _ _GFP_WAIT flag is not set, the inner loop is collapsed, and each node in the list is tried in sequence with no retries. So given the node list {3,3,1} from the example above, alloc_pages_nodelist() attempts allocation from the nodes in the following order before finally failing: 3 3 1.

Kernel API's

Allocating Whole Pages, alloc_pages_nodelist()

Device drivers or other kernel code that wish to allocate whole memory pages from a specific node can call alloc_pages_nodelist() directly. If the caller has a list of mnemonic strings, it must first convert the strings to a node list with memtypes_to_nodelist() before calling alloc_pages_nodelist().

For sake of speed in allocating pages during page faults, alloc_pages_nodelist() does not call check_nodelist() to check the validity of the passed node list. Instead, it does the following (refer to the flow chart above):

- if depth is greater than MAX_NR_NODES, fail immediately (return NULL).
- in the inner loop, if the current node ID in the list is invalid, skip this entry and move on to the next ID in the list.

Note that the passed node list will never be invalid if alloc_pages_nodelist() was called as a result of a page fault or a slab allocation, because kmalloc_nodelist(), do_mmap_nodelist(), and do_brk_nodelist() all check the validity of the list beforehand.

Slab Allocator, kmalloc_nodelist()

Device drivers or other kernel code that wish to allocate memory of arbitrary size from a specific node can make use of a new interface to the slab allocator, kmalloc_nodelist(), which takes as an extra argument a pointer to a struct node_list object. It's prototype is as follows:

```
void * kmalloc_nodelist (struct node_list * nl, size_t size, int flags);
```

There is also a new slab interface that allows creation of a new cache that includes a node list:

```
kmem_cache_t *  kmem_cache_create_nodelist (struct node_list * nl,
        const char *name, size_t size, size_t offset, unsigned long flags,
            void (*ctor)(void*, kmem_cache_t *, unsigned long),
            void (*dtor)(void*, kmem_cache_t *, unsigned long));
```

The new cache can then be used when allocating objects by passing it to kmem_cache_alloc(). The new objects will be allocated from the nodes listed in the cache objects node list.

Both of these new methods perform the following checks on the passed node list:

- if the node list pointer is NULL, or the list is empty, the new slab object or cache will not have any node preference.
- if the node list is invalid as indicated by check_nodelist(), both methods fail, returning NULL.

do_mmap_nodelist() and do_brk_nodelist()

Kernel code that wishes to create new mappings for a process can call do_mmap_nodelist() or do_brk_nodelist() directly. The current prototypes are identical to the original do_mmap() and do_brk(), with the addition of a node_list pointer as the last argument.

If the passed node_list pointer is non-NULL and enabled (depth is non-zero), but the list is invalid as indicated by check_nodelist(), the mapping fails, and both methods return -EINVAL.

User API’s

Mmap_memtypes() and brk_memtypes()

These new system calls are implemented to allow creating memory maps from user space with node information. They essentially provides user-level access to the kernel methods do_mmap_nodelist() and do_brk_nodelist(). The prototypes are the same as the current system calls, with two additional arguments:

```
void * mmap_memtypes(void *start, size_t length, int prot,
    int flags, int fd,  off_t offset, char * memtypes,
    int memtypes_len);

int brk_memtypes(void *end_data_segment, char * memtypes,
    int memtypes_len);
```

The memtypes argument is a pointer to a user buffer holding a packed list of null-terminated strings. The strings represent the memory type mnemonics, and their order in the buffer is the order of node preference for the region. The memtypes_len argument is the total size of the user buffer in bytes.

Note that these new libc functions are not reserved by the POSIX standard. Applications that use them have to be compiled with -D_GNU_SOURCE.

The new syscalls are also used by the dynamic linker (ld.so) in MTA-aware glibc, to create the maps for a program’s shared libraries. The following checks are made on the arguments passed to mmap_memtypes() and brk_memtypes():

- If the memtypes buffer pointer is NULL, or if memtypes_len is zero, the new mapping created will not have any node list preference, i.e. it will be as if the regular mmap() and brk() syscalls were used.
- If the copy of the user buffer to kernel space fails (for instance the memtypes pointer is invalid), the mapping fails.
- There is an upper limit of one page (4096 bytes) on the user buffer size. If memtypes_len is greater than PAGE_SIZE, the mapping fails.
- If any of the memory type mnemonic names in the memtypes buffer do not match any of the kernel’s node names, the new mapping created will not have any node list preference.
- The usual conditions exist on the remaining arguments (for instance, for a file mapping the file descriptor must refer to a valid open file).

/proc Interface

There are two new entries in the /proc file system.

/proc/nodeinfo

The first is /proc/nodeinfo, which lists the node configuration of the kernel, including the name, physical address range, and whether default page allocation is allowed, of each configured node.

/proc/[pid]/nodemap

The second is an extension of the Memory Accounting tool. If the kernel config option CONFIG_MEMORY_ACCOUNTING is enabled along with CONFIG_MEMTYPE_ALLOC, a new proc entry, /proc/[pid]/nodemap will be available. The information is similar to the Memory Accounting Tool’ s /proc/[pid]/memmap, except that instead of displaying the page

usage counter for every resident page in each region, the node ID of resident pages are displayed. Pages for a region that are not yet resident are shown with a dash character "-".

In other words, for every line (region) printed by /proc/[pid]/maps, /proc/[pid]/nodemap also prints a line, showing the node ID of resident pages for that region.

Tracing MTA with Linux Trace Toolkit

Important MTA events are captured by the run-time creation of Linux Trace Toolkit (LTT) custom events for MTA. The following events are defined in include/linux/vmnode.h, and are called at the appropriate locations in the kernel where the corresponding events occur:

- TRACE_MTA_ELF_MEMTYPES

An ELF executable or ld.so was loaded containing a .memtypes NOTE section.

- TRACE_MTA_MMAP_MEMTYPES

Entry to mmap_memtypes system call with a non-empty memtypes buffer.

- TRACE_MTA_BRK_MEMTYPES

Entry to brk_memtypes system call with a non-empty memtypes buffer.

- TRACE_MTA_MMAP_NODELIST do_mmap_nodelist() was called with a valid node list.
- TRACE_MTA_BRK_NODELIST do_brk_nodelist() was called with a valid node list.
- TRACE_MTA_KMALLOC_NODELIST kmalloc_nodelist() was called with a valid node list.
- TRACE_MTA_KMEM_CACHE_CREATE_NODELIST kmem_cache_create_nodelist() was called with a valid node list.

- TRACE_MTA_SLAB_ALLOC

A group of contiguous pages were allocated for a slab cache object containing a node list.

- TRACE_MTA_VMA_ALLOC

A page was allocated for a copy-on-write, for an anonymous or file mapping containing a node list.

- TRACE_MTA_PAGE_CACHE_ALLOC

A page was allocated and placed in the page cache, for a file mapping containing a node list.

With these events, it’s possible to trace MTA-related activity from the time a program was loaded, to the creation of its memory map, down to the allocation of memory pages for the program. The events can also trace the creation of new slab caches containing node lists, down to allocation of pages for the cache objects.

Additional Information

Porting MTA to other Architectures

At this time, only the ARM OMAP1510 Innovator platform has MTA support. To port MTA to other architectures:

- First of all, the architecture must support discontiguous memory.
- Add the CONFIG_MEMTYPE_ALLOC option to arch/[arch]/config.in if CONFIG_DISCONTIGMEM is defined. See arch/arm/config.in for example.

- Add system call entry points for sys_brk_memtypes() and old_mmap_memtypes() and define their syscall numbers. See arch/arm/kernel/calls.S and include/asm-arm/unistd.h for example.

- Implement old_mmap_memtypes() (sys_brk_memtypes() is implemented in generic kernel code in mm/mmap.c). See arch/arm/kernel/sys_arm.c for example implementation.
- Configure the system's memory nodes using the start and end physical addresses of each node in the mta_nodes[] array. How discontiguous memory nodes are initially configured is very architecture specific. See include/asm-arm/arch-omap1510/memory.h, arch/arm/mach-omap1510/innovator.c, and arch/arm/mm/init.c for an example of how this is done for ARM and the Innovator platform.

Limitations

- In ELF binaries, the first file page offset of the initialized data segment is usually the same file page offset as the last page of text (the end of text and start of data share the same page). Because of this, the same allocated page frame in the kernel's page cache is shared between the last page of text and the first page of initialized data. Therefore, if the program references the last page of text after it references the first page of data (which is usually the case), the last page of the text region will be located in the node of the data region, not in the text's node.
- The Innovator's SRAM is very small, and page allocations from SRAM will begin to fail very quickly. The text segment of ld.so happens to just barely fit in SRAM. Even then, the kernel will attempt to allocate a cluster of pages for a region instead of only one during a file mapping page fault, and if that many pages are not free in SRAM, the cluster allocation will fail.

Future Enhancements

- Expand maximum allowable nodes beyond 16.
- Allow separation of data/bss/brk/stack segments into different nodes.
- For native elfmemtypes tool, check mnemonic names against /proc/nodeinfo.

Notes

Source Code

linux-mta-041004.tar.bz2 is a kernel soruce archive including MTA. Please someone isolate MTA funtion from the tarball. MTA util and mta-glibc-2.2.5.patch are also available.

Category:

- Memory Type Based Allocation

> From: eLinux.org

Runtime Memory Measurement

This page has a collection of ideas and resources having to do with measuring runtime memory of a Linux system.

Unfortunately, the existing memory measurement techniques do not give a 100% accurate accounting of memory pages (since some pages are counted more than once by some measures). See Accurate Memory Measurement

- that page describes techniques (and patches) which can be used to measure the runtime memory more accurately.

Contents

Measuring memory in Linux (the basics)

Here are some basic techniques for measuring memory usage in Linux.

'free' and /proc

The 'free' command shows the memory on a machine, in certain categories.

[need explanation of categories here...'man free' doesn't explain the numbers]

```
$ free
             total       used       free     shared    buffers     cached
Mem:        507564     481560      26004          0      68888     185220
-/+ buffers/cache:     227452     280112
Swap:      2136604     105168    2031436
```

This information is obtained from /proc/meminfo, which has additional details not shown by the 'free' command.

The following is on my machine with 512 Mb RAM, running Linux 2.6.3:

```
$ cat /proc/meminfo
MemTotal:       507564 kB
MemFree:         26004 kB
Buffers:         68888 kB
Cached:         185220 kB
SwapCached:      29348 kB
Active:         342488 kB
Inactive:        32092 kB
HighTotal:           0 kB
HighFree:            0 kB
LowTotal:       507564 kB
LowFree:         26004 kB
SwapTotal:     2136604 kB
SwapFree:      2031436 kB
Dirty:              88 kB
Writeback:           0 kB
Mapped:         165648 kB
Slab:            73212 kB
Committed_AS:   343172 kB
PageTables:       2644 kB
VmallocTotal:   524212 kB
VmallocUsed:      7692 kB
VmallocChunk:   516328 kB
```

See http://lwn.net/Articles/28345/ for a description of these fields

Measuring user process memory use

'ps' fields for memory information

The 'ps' command provides information about the memory usage of processes on a Linux system. However, it is not well documented. Here are some notes on using 'ps' and /proc to view memory usage information on a running Linux system:

meaning of ps fields:

- %Mem - percent of memory
- VSZ - Virtual Size
- RSS - Resident Set Size
- SIZE - Equivalent to VSZ
- others??

'top' fields for memory information

See 'man top':

- %MEM -- Memory usage (RES)
  -
    - A task's currently used share of available physical memory.
- VIRT -- Virtual Image (kb)
  -
    - The total amount of virtual memory used by the task. It includes all code, data and shared libraries plus pages that have been swapped out.
    - VIRT = SWAP + RES
- SWAP -- Swapped size (kb)
  -
    - The swapped out portion of a task's total virtual memory image.
- RES -- Resident size (kb)

  -
    -
      - The non-swapped physical memory a task has used.
      - RES = CODE + DATA.
- CODE -- Code size (kb)
  -
    -
      - The amount of physical memory devoted to executable code, also known as the 'text resident set' size or TRS
- DATA -- Data+Stack size (kb)
  -
    -
      - The amount of physical memory devoted to other than executable code, also known as the 'data resident set' size or DRS.
- SHR -- Shared Mem size (kb)
  -
    -
      - The amount of shared memory used by a task. It simply reflects memory that could be potentially shared with other processes.
- nDRT -- Dirty Pages count
  -
    -
      - The number of pages that have been modified since they were last written to disk. Dirty pages must be written to disk before the corresponding physical memory location can be used for some other virtual page.

Are the following assertions true:??

- virtual memory usage of a process, excluding shared libs = VIRT - SHR
- physical memory usage of a process excluding shared libraries = RES
  - SHR

/proc info

see 'man proc' for detailed information about the files and fields in the /proc filesystem.

/proc/\/statm

/proc/\/statm fields: columns are (in pages):

<table>
<tr><td>total program size|</td></tr>
<tr><td>resident set size|</td></tr>
<tr><td>shared pages|</td></tr>
<tr><td>text (code) |</td></tr>
<tr><td>data/stack |</td></tr>
<tr><td>library |</td></tr>
<tr><td>dirty pages |</td></tr>
</table>

Here an example: 693 406 586 158 0 535 0

/proc/\/status

/proc/\/status fields:

- Vm Size: 2772 kB
- Vm Lck: 0 kB - ???
- Vm RSS: 1624 kB
- Vm Data: 404 kB
- Vm Stk: 24 kB
- Vm Exe: 608 kB
- Vm Lib: 1440 kB

/proc/\/maps

The process maps shows the actual memory areas that have been mapped into a process' address space, and their permissions.

Example:

```
$ cat /proc/25042/maps
08048000-080e0000 r-xp 00000000 03:05 196610     /bin/bash
080e0000-080e6000 rw-p 00097000 03:05 196610     /bin/bash
080e6000-08148000 rwxp 00000000 00:00 0
40000000-40016000 r-xp 00000000 03:05 147471     /lib/ld-2.3.3.so
40016000-40017000 rw-p 00015000 03:05 147471     /lib/ld-2.3.3.so
40017000-40018000 rw-p 00000000 00:00 0
40018000-40019000 r--p 00000000 03:05 184090     /usr/share/locale/en_US/LC_IDENTIFICATION
40019000-4001a000 r--p 00000000 03:05 184089     /usr/share/locale/en_US/LC_MEASUREMENT
4001a000-4001b000 r--p 00000000 03:05 184083     /usr/share/locale/en_US/LC_TELEPHONE
4001b000-4001c000 r--p 00000000 03:05 184091     /usr/share/locale/en_US/LC_ADDRESS
4001c000-4001d000 r--p 00000000 03:05 184086     /usr/share/locale/en_US/LC_NAME
4001d000-4001e000 r--p 00000000 03:05 184084     /usr/share/locale/en_US/LC_PAPER
4001e000-4001f000 r--p 00000000 03:05 184088     /usr/share/locale/en_US/LC_MESSAGES/SYS_LC_MESSAGES
4001f000-40020000 r--p 00000000 03:05 184087     /usr/share/locale/en_US/LC_MONETARY
40020000-40026000 r--p 00000000 03:05 183689     /usr/share/locale/ISO-8859-1/LC_COLLATE
40026000-40027000 r--p 00000000 03:05 184082     /usr/share/locale/en_US/LC_TIME
40027000-4002a000 r-xp 00000000 03:05 147459     /lib/libtermcap.so.2.0.8
4002a000-4002b000 rw-p 00002000 03:05 147459     /lib/libtermcap.so.2.0.8
4002b000-4002c000 rw-p 00000000 00:00 0
4002c000-4002e000 r-xp 00000000 03:05 147482     /lib/libdl-2.3.3.so
4002e000-4002f000 rw-p 00001000 03:05 147482     /lib/libdl-2.3.3.so
4002f000-40171000 r-xp 00000000 03:05 147511     /lib/tls/libc-2.3.3.so
40171000-40174000 rw-p 00142000 03:05 147511     /lib/tls/libc-2.3.3.so
40174000-40177000 rw-p 00000000 00:00 0
40177000-40178000 r--p 00000000 03:05 184085     /usr/share/locale/en_US/LC_NUMERIC
40178000-401a4000 r--p 00000000 03:05 183688     /usr/share/locale/ISO-8859-1/LC_CTYPE
401a4000-401a5000 r-xp 00000000 03:05 180462     /usr/lib/gconv/ISO8859-1.so
401a5000-401a6000 rw-p 00001000 03:05 180462     /usr/lib/gconv/ISO8859-1.so
401b3000-401bd000 r-xp 00000000 03:05 147492     /lib/libnss_files-2.3.3.so
401bd000-401be000 rw-p 00009000 03:05 147492     /lib/libnss_files-2.3.3.so
bfffa000-c0000000 rwxp ffffb000 00:00 0
ffffe000-fffff000 ---p 00000000 00:00 0
```

mem_usage command to consolidate data

David Schleef wrote a program to consolidate the information from /proc/\/maps, and total up each kind of memory for a process.

Here it is: Media:mem_usage (It was obtained from http://www.schleef.org/~ds/mem_usage)

Here is the result of running mem_usage on the process used in the previous example:

```
$ ./mem_usage 25042
Backed by file:
  Executable                r-x  2048
  Write/Exec (jump tables)  rwx  0
  RO data                   r--  240
  Data                      rw-  56
  Unreadable                ---  0
  Unknown                        0
Anonymous:
  Writable code (stack)     rwx  416
  Data (malloc, mmap)       rw-  20
  RO data                   r--  0
  Unreadable                ---  4
  Unknown                        0
```

Inaccuracies of kernel reporting mechanisms

Many of the memory reporting mechanisms for the kernel are inaccurate, due to not recording sufficient information about the true state of the system. Here are some random notes on these inaccuracies. To see information on different methods of getting more accurate memory information, see Accurate Memory Measurement

- "copy-on-write" pages - an mmap'ed file may be very large in the process address space, but empty until written to.

From Ratboy on Slashdot:

```
The mmap() call can map a file (backing store) and allow data to be shared. Memory does
not need to be used until the data is read (or written). And this time, the backing
store doesn't even need swap (because the file is the backing store).

...

A page of code that is shared - may become a page of code that is private. A page of
data that is unwritten doesn't have to exist. Even if it is read! A page of data that
is written may STILL be shared.
```

From others on Slashdot:

Top will show you the same as ps does, ps reads /proc/\/statm and asks what's going on. The problem on linux is the copy on write principle which saves heaps of memory, but makes it virtually impossible to figure out what belongs to what. The thing is, when you fork it maps the memory and marks everything as copy on write, when something needs to write to part of the memory, then it will make the copy for each process.

However asking the process how much memory it has allocated will show all memory including stuff that is marked copy on write - that is, I could have 100 processes showing they each use 1.4MB of memory, because they all share the same library, but in fact, its the same copy they are all using so I'm only using 1.4 MB instead of 140MB (+PCB et. al)

Each thread in a process shows up as consuming the same amount of memory (either this only happens under Linuxthreads or I don't have any threaded applications running on my system).

Device mappings show up as consumed memory (which generates plenty of complaints about the X server). If you want to find out how much memory X is actually using (bytes in cached pixmaps on behalf of each process and sans device mappings), try the program here: http://69.142.116.122/dist/pixmap_mem-1.0.tgz

This contains a tiny program that lists how much memory X is using for other programs by caching pixmaps and a perl script that lists how much memory X is using sans device mappings. }}}

- pmap is a utility which shows the memory usage of a process (it looks like it just reads and interprets /proc/\/maps).

Someone on Slashdot said:

```
pmap *also* overestimates memory usage, because some portion of the mapped address
space isn't actually in use. RSS, on the other hand, only measures memory that is
actually in use, but doesn't distinguish between memory that is shared and memory
that is not. VSZ is the most pessimistic measure, since it includes all mapped
memory, shared and unshared.
```

Heap memory usage

Heap is the dynamically allocated memory inside each process' address space that is managed by the application itself. The structure of this memory is actually managed by the C library, with the application calling malloc() and free().

glibc has the capability to collect statistics information of heap functions like malloc() and other functions like memory leak checking or double free.

Memory Debuggers

Several tools are available to analyze memory allocations, watch for reading and writing beyond the end of allocated memory, and do other tasks which help with debugging and tuning memory operations of a program. See Memory Debuggers for a list of different tools and their features.

Measuring kernel memory use

Kernel Stack Usage

- Tim is adding a stack checking function to KFT (See Kernel Function Trace)
  -
    - This new feature has not yet been published
- Recent -mm tree added stack-corruption-detector.patch (8th March, 2006)
- The scripts/checkstack.pl script in the kernel tree will show the functions with the largest static stack footprint.
- Enabling CONFIG_DEBUG_STACKOVERFLOW in the kernel will enable checking for low-stack situations in the irq handler.
- Enabling CONFIG_4KSTACKS will cause stack overflows to occur more frequently, particularly with certain kernel code such as XFS enabled.

General kernel memory use

- http://www.halobates.de/memorywaste.pdf
  - Great paper by Andi Kleen, of SUSE Labs, about dynamic memory usage of Linux systems
- Check /proc/slabinfo to find how much memory is being used by the kernel SLAB allocator (or SLUB or SLOB, depending on what is enabled).

Kernel memory analysis tools and project

The Linux Foundation CE Workgroup has a project to analyze the kernel's dynamic memory utilization. See: Kernel dynamic memory analysis

Category:

- System Size

> From: eLinux.org

Tims Notes on ARM memory allocation

This page has a collection of notes that I made while working on the stack limit patches for Sony. This was a set of patches that tried to map kernel memory areas as 4K pages, so that an individual page of the stack could be unmapped, and a page fault generated, when stack space was running low. (This was related to testing an implementation of 4K stacks on ARM).

These notes are placed here in the hopes that they will be useful for someone working on ARM memory management.

Contents

startup memory

At start up the kernel (usually) automatically determines the physical memory areas. However, the user can manually specify one or more physical memory areas using "mem=..." kernel command line options. When used, these override any automatically determined values. These are parsed by early_mem().

The initial description of physical memory regions are stored in the global 'meminfo' structure, and each region is described as a 'bank'. Some initial physical memory is utilized by the bootmem allocator. It is from this pool of physical memory, that the page tables are built, which allows the MMU to be turned on, and the kernel switched over to virtual memory. Once this process is done, the bootmem pool is freed and all system pages are turned over to the various page and slab allocators of the system. A good reference for this is Mel Gorman's excellent book on the topic: Understanding the Linux Virtual Memory Manager The prior link is to the PDF, here's a link to html: http://www.kernel.org/doc/gorman/html/understand/ Chapter 5 talks about the bootmem allocator.

page tables

The page tables are, unsurprisingly, initialized by paging_init(). ARM uses a somewhat weird way of mapping the Linux page tables onto the ARM hardware tables. This method is described in comments in the file arch/arm/include/asm/pgtable.h, with additional macros defined in pgtable-hwdef.h and page.h. Basically, Linux supports 4 levels (pgd, pud, pmd, and pte), and ARM maps this onto 2 levels (pgd/pmd and pte). The nomenclature in the code is hard to follow, because Linux generic code thinks that pgd is the top level of page tables, but internally the ARM code uses pmd macros to refer to the top hardware page table.

Originally, Linux used 4KB mappings for ARM, but they have converted over to mostly 1MB mappings (at least for the Linux kernel). According to my colleague, Frank Rowand, bad things happen if a physical page is represented in the page table by more than one entry (for example, if a physical page has both an entry as a small page in a second-level page table, and is inside a region covered by a large "section" page entry in a first-level page table.

At the hardware level, ARM supports two page table trees simultaneously, using the hardware registers TTBR0 and TTBR1. A virtual address is mapped to a physical address by the CPU depending on settings in TTBRC. This control register has a field which sets a split point in the address space. Addresses below the cutoff value are mapped through the page tables pointed to by TTBR0, and addresses above the cutoff value are mapped through TTBR1. TTBR0 is unique per-process, and is in current->mm.pgd (That is, current->mm.pgd == TTBR0 for that process). That is, when a context switch occurs, the kernel sets TTBR0 to the current->mm.pgd for new process. TTBR1 is global for the whole system, and

represents the page tables for the kernel. It is referenced in the global kernel variable swapper_pg_dir. Note that both of these addresses are virtual addresses. You can find the physical address of the first-level page table by using virt_to_phys() functions on these addresses.

I found the following ARM reference material helpful in trying to understand the page table layout: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0333h/Babbhigi.html

- this is the chapter on the MMU in the ARM1176JZ-S Technical Reference Manual. In particular, diagram 6.9 showing ARMv6 section, supersection and page translation, in section 6.11.2 of the manual is very useful. It is here.

Note that in particular, the second-level page table (PTE) layout is weird. The ARM hardware supports 1K tables at this level (256 entries of 4bytes each). However, the Linux kernel needs some bits that are not provided on some hardware (like DIRTY and YOUNG). These are synthesized by the ARM MMU code via permissions and faulting, effectively making them software-managed. They have to be stored outside the hardware tables, but still in a place where the kernel can get to them easily. In order to keep things aligned into units of 4K, the kernel therefore keeps 2 hardware second-level page tables and 2 parallel arrays (totalling 512 entries) on a single page. When a new second-level page table is created, it is created 512 entries at a time, with the hardware entries at the top of the table, and Linux software entries (synthesized flags and values) at the bottom.

The kernel has a mixture of 1MB mappings and 4KB mappings. This relieves some pressure on the TLB, which is used by the CPU to cache virtual-to-physical address translations.

paging_init()

The page tables and paging infrastructure are initialized as follows:

- paging_init() is called by setup_arch() after the meminfo structure has been initialized and the bootmem allocator is ready. It calls the following routines:
  - memblock_set_current_limit()
  - build_mem_type_table() - builds a table of memory types. This has the page protection flags that are available for different memory types, for the current ARM processor. Different ARM processors have changed what flags they use, and where they are located in the page table entries, over the years. the 'mem_types' table encapsulates the settings for the running processor.
  - prepare_page_table() - this zeros out certain areas of the first-level page table (called pmd in this routine). For example, it zeros out the areas of the page table that will be covered by user-space (areas below the start of the kernel address space).
  - map_lowmem() - create the memory mappings (page table entries) for the lower portions of kernel memory. This is the "normal" memory that will be used by the kernel for static code and data, stack, and regular dynamic allocations.
  - devicemaps_init() - create the memory mappings for special CPU areas (e.g. cache flushing regions, and interrupt vectors) and reserved IO areas in the memory map.
  - kmap_init() - create the memory mapping for highmen ('pkmap')

A call tree for a regular page table setup is:

```
start_kernel()
  setup_arch()
    paging_init()
      map_lowmem()
        create_mapping()
          alloc_init_pud() - for a range of pgd entries
            alloc_init_section() - for a range of "pud" entries
              *pmd = __pmd(<stuff>) - actually set the pmd/pgd entries for a SECTION mapping
              or
              alloc_init_pte()
                early_pte_alloc() - to get a page for the pte table
                  pte_offset_kernel() - get address of 'linux' portion of pte table page
                set_pte_ext(<stuff>) - create the individual PTE entries, for a range of entries
              =>  cpu_set_pte_ext() - macro to wrapper calling through a cpu-specific routine
              =>  processor.set_pte_ext() - function pointer to cpu-specific routine
              =>  cpu_armXXX_set_pte_ext() or cpu_procvX_set_pte_ext - cpu-specific-routine
```

set_pte_ext ('set page table entry extended') takes the following arguments:

- pte - pointer to pte entry to change (the address of the linux entry
  - not the 'hardware' entry)
- value - the value to place in the entry - this is usually created with the macro pfn_pte(x,y), where x has the page frame number (physical address of the page for this mapping), and y represents the protection flags for the page. This should be the 'linux' flags for the page.
- value2 - the value to place in the "extended" or "extra" entry

Note that the assembly routine for this, which is cpu-specific, will take the Linux PTE values and derive appropriate hardware mappings for the hardware PTE entry.

Nomenclature

- pgd = page global directory
- pud = page upper directory
- pmd = page middle directory
- pte = page table entry (this is confusion, 'pte' can refer to both the table and an entry in the table)
  - ptep = pointer to pte entry
- pfn = page frame number = the base address (no bottom bits) of a physical page address

> From: eLinux.org

Resource Management

This page has information about various kernel based resource management frameworks which are of interest to embedded developers.

Contents

Open Source Projects

- CKRM, Class-based Kernel Resource Management (ckrm-tech mailing list and archives).
- UBC, OpenVZ User beancounters.
- OpenSourceMID.org Opensourcemid

CKRM/RG

- Kernel Summit: Class-based Kernel Resource Management - CKRM LWN article from Kernel Summit 2004.
- Resource Groups - the second coming of CKRM, heavily remodelled.

UBC/Beancounters

- Initial patch and subsequent thread on linux-kernel.
- Resource Beancounters LWN article.

Additional Resources

- Resource Management - Infrastructure choices thread on linux-kernel, attempting to focus on infrastructure convergence using a simple basic model that more robust implementations can be built on top of.

Category:

- Resource Management

> From: eLinux.org

Device drivers

Manuals

- Linux kernel internals reference, wikibook - under construction
- Linux Device Drivers, 3rd Edition
- Tutorial for writing parallel port driver

Sample drivers

- LDT - Linux Driver Template - sample template of Linux device driver for learning and starting source for a custom driver. Implements UART char device driver for example. Uses following Linux facilities: module, platform driver, file operations (read/write, mmap, ioctl, blocking and nonblocking mode, polling), kfifo, completion, interrupt, tasklet, work, kthread, timer, misc device, proc fs, UART 0x3f8, HW loopback, SW loopback, ftracer. The code is in working condition and runs with test script.
- LDD3 - Samples for boot Linux Device Driver, 3rd edition, updated, compiled with kernel 3.2.0
  - pci_skel.c
    - PCI skeleton
  - sbull.c
    - simple block device
  - scull
    - simple char device
  - snull.c
    - simple network device
- vivi.c - Virtual Video driver, uses V4L2
  - works
- mem2mem_testdev.c - virtual v4l2-mem2mem example device driver
- usb-skeleton.c - USB driver skeleton (can be compiled with trivial fix)
- skeletonfb.c - Frame Buffer device skeleton (can't be compiled)
- pcihp_skeleton.c - PCI Hot Plug Controller Skeleton Driver
- loopback.c - simple net_device implementing ifconfig lo
- gpio_driver - simple GPIO driver for Raspberry Pi model B+ (not fully tested yet)

Resources

- Device Tree - information about device tree (increasingly required for new embedded drivers)

> From: eLinux.org

Device Tree

Contents

Introduction

Device Tree data can be represented in several different formats. It is derived from the device tree format used by Open Firmware to encapsulate platform information and convey it to the Linux operating system. The device tree data is typically created and maintained in a human readable format in .dts source files and .dtsi source include files. The Linux build system pre-processes the source with cpp.

The device tree source is compiled into a binary format contained in a .dtb blob file. The format of the data in the .dtb blob file is commonly referred to as a Flattened Device Tree (FDT). The Linux operating system uses the device tree data to find and register the devices in the system. The FDT is accessed in the raw form during the very early phases of boot, but is expanded into a kernel internal data structure for more efficient access for later phases of the boot and after the system has completed booting.

Currently the Linux kernel can read device tree information in the ARM, x86, Microblaze, PowerPC, and Sparc architectures. There is interest in extending support for device trees to other platforms, to unify the handling of platform description across kernel architectures.

The Flattened Device Tree is...

The Flattened Device Tree (FDT) is a data structure. Nothing more.

It describes a machine hardware configuration. It is derived from the device tree format used by Open Firmware. The format is expressive and able to describe most board design aspects including:

- the number and type of CPUs,
- base addresses and size of RAM,
- busses and bridges,
- peripheral device connections, and
- interrupt controllers and IRQ line connections.

Just like initrd images, an FDT image can either be statically linked into the kernel or passed to the kernel at boot time.

The Flattened Device Tree is not...

- is not a solution to all board port problems
  - Nothing will eliminate all board specific drivers for custom and complex boards.
- is not a firmware interface
  - It might be part of a generic firmware interface, but on its own the device tree is just a data structure.
  - does not replace ATAGS... but an FDT image can be passed via an ATAG.
  - See "Competing Solutions" below
- is not intended to be a universal interface.
  - It is a useful data structure which solves several problems, but whether or not to use it is still up to the board port author.
- is not an invasive change
  - ~~No requirement to use FDT approach in a board port~~
  - Device Tree is required for new board support in the ARM architecture.
  - No requirement to convert existing board ports
  - No requirement to modify existing firmware

History

- How device tree got into Linux and how it has evolved

Future

- How device tree is changing and where it is headed

Advantages

for distributions

- potentially fewer kernel images needed on an installer image (ie. for ARM netbooks)
  - Ship one FDT image per machine (\<4k/machine) instead of 1 kernel image per machine (~1-2MB/machine) with a small number of sub-arch kernel images (ie. ARM11, CortexA8, CortexA9, etc).
  - Becomes feasible for current installer image to boot on future hardware platforms using same chipset.
  - Note: FDT is only part of the solution here. Some boot software is still required to select and pass in the correct FDT image.

for System on Chip (SoC) vendors

- Reduce or eliminate effort needed to write machine support code (ie arch/arm/mach-*). Focus on device driver development instead.

for board designers

- Reduce effort required to port.
  - SoC vendor supplied reference design binaries may also be bootable on custom machine.
- No need to allocate a new global ARM machine id for each new board variant.
  - Use the device tree \,\ namespace instead
- Most board specific code changes constrained to device tree file and device drivers.
- Example: Xilinx FPGA toolchain has a tool to generate a device tree source file from the FPGA design files.
  - Since the hardware description is constrained to the device tree source, FPGA engineers can test design changes without getting involved with kernel code.
  - Alternately, kernel coders don't need to manually extract design changes from the FPGA design files.

for embedded Linux ecosystem

- smaller amount of board support code to merge
- greater likelyhood of mainline support for boards from "uninterested" vendors
- greater ability to correct poor board support by fixing or replacing broken FDT images.

for firmware/bootloader developers

- reduce impact of getting board description wrong (FDT stored as a separate image instead of statically linked into firmware). If initial release gets the board description wrong, then it is easily updated without a risky reflash of firmware.
- expressive format to describe related board variants without allocating new machine numbers or new ATAGs.
- Note: The FDT isn't a replacement to ATAGS, but does supplement them.

Other advantages

- Device tree source and FDTs can easily be machine generated and/or modified.
  - Xilinx FPGA tools do device tree source generation
  - U-Boot firmware can inspect and modify an FDT image before booting

Competing Solutions

board specific data structures

Some platforms use board-specific C data structures for passing data from the bootloader to the kernel. Notable here is embedded PowerPC support before standardizing on the FDT data format.

Experience with PowerPC demonstrated that using a custom C data structure is certainly an expedient solution for small amounts of data, but it causes maintainability issues in the long term and it doesn't make any attempt to solve the problem of describing the board configuration as a whole. Special cases tend to grow and there is no way for the kernel to determine what specific version of the data structure is passed to it. PowerPCs board info structure ended up being a mess of #ifdefs and ugly hacks, and it still only passed a handful of data like memory size and Ethernet MAC addresses.

ATAGs have the elegance of providing an well defined namespace for passing individual data items (memory regions, initrd address, etc) and the operating system can reliably decode them. However, only a dozen or so ATAGs are defined and is not expressive enough to describe the board design. Using ATAGs essentially requires a separate machine number to be allocated for each board variant, even if they are based on the same design.

That being said, an ATAG is an ideal method for passing an FDT image to the kernel in the same way an ATAG is used to pass the initrd address.

ACPI

Firmware providing the Advanced Configuration and Power Interface exports a hardware description in the form of the Differentiated System Description Table (DSDT). ACPI is found on x86 compatible systems and has its roots in the original IBM PC BIOS.

UEFI

The Extensible Firmware Interface is an interface specification for passing control from a platforms firmware to the operating system. It was designed by Intel as a replacement for the PC BIOS interface.

ARM holdings is a member of the United EFI Forum. It is conceivable that there will be an ARM implementation of UEFI.

Open Firmware

Open Firmware is a firmware interface specification designed by Sun in the late 1980's, and ported to many architectures. It specifies a runtime OS client interface, an cross platform device interface (FCode), a user interface, and the Device Tree layout for describing the machine.

FDT is to Open Firmware what DSDT is to ACPI. The FDT reuses Open Firmware's established device tree layout. In fact, Linux PowerPC support uses the same codebase to support both Open Firmware and FDT platforms.

Some Notes on the Competing Solutions

Most of the competing solutions listed above provided feature rich firmware interfaces including both machine description and runtime services. Conversely, the FDT is only a data structure and doesn't specify any firmware interface details. Board ports using the FDT are typically booted from simple firmware implementations like U-Boot and don't provide any form of runtime services.

A common design goal of the feature rich firmware interfaces is to provide an abstract boot interface that factors away the differences between different hardware platforms, at least enough for the OS to initialize its own native device drivers. The idea is to be able to boot 'old' OS images on 'new' hardware, like how a Linux LiveCD image doesn't have explicit knowledge of the hardware configuration, but relies on the information provided to it by firmware.

Typical design goals for embedded firmware is to a) boot the OS as quickly as possible, b) upgrade the OS image, and maybe c) provide some low level debug support during initial board bringup. Focus tends to shift away from firmware once the OS is bootable since the kernel drivers the hardware directly (doesn't depend on firmware runtime services). In fact, firmware updates are discouraged due to the risk of rendering a board unbootable. ACPI, UEFI and OpenFirmware solutions, while arguably 'better', often don't boot as fast, and are more complex than required by the embedded system. In this regard the FDT approach has the advantage due to its simplicity. ie. the FDT provides an equivalently expressive way to describe hardware, but it works with existing firmware and can be updated without reflashing firmware.

Resources

Wiki and in-kernel documentation

The main device Tree wiki is at: http://www.devicetree.org/Main_Page

```
http://www.devicetree.org/Device_Tree_Usage is an excellent introduction to device tree
concepts and the representation of those concepts in device tree source.  (But check the
"last modified" date at the bottom of the page (currently 23 October 2010) when comparing
to other resources.)
```

Documentation about device tree is available in the Linux kernel Documentation directory: Documentation/devicetree. See https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/Documentation/devicetree

Some especially useful files are:

- ABI.txt: comments on stable binding and general bindings rules
- resource-names.txt: -name properties containing an ordered list of names corresponding to another property
- usage-model.txt: information about different elements of bindings
- vendor-prefixes.txt: vendor prefix registry
- the bindings directory has details about the syntax and expected elements for each device type representable in the dts and used by kernel frameworks and drivers
- bindings/submitting-patches.txt: important details for
  - patch submitters
  - kernel maintainers

FAQ, tips and/or best practices

See the Linux Drivers Device Tree Guide.

Presentations, Papers and Articles

- Solving Device Tree Issues, LinuxCon Japan 2015 by Frank Rowand
- "The Device Tree as a Stable ABI: A Fairy Tale?", ELC 2015 by Thomas Petazzoni
  - http://elinux.org/images/0/0a/The_Device_Tree_as_a_Stable_ABI-_A_Fairy_Tale%3F.pdf
- "Transactional Device Tree & Overlays: Making Reconfigurable Hardware Work", ELC 2015 by Pantelis Antoniou
  - PDF
  - YouTube video
- "Device Tree for Dummies", ELC 2014 by Thomas Petazzoni
  - PDF
  - YouTube video
- Device trees I: Are we having fun yet? - Neil Brown, LWN.net November 2013
- Device trees II: The harder parts - Neil Brown, LWN.net November 2013
- "Device Tree for Dummies", ELC Europe 2013 by Thomas Petazzoni
  - PDF
  - YouTube video
- "Transactional Device Tree & Overlays: Making Reconfigurable Hardware Work", ELC Europe 2014 by Pantelis Antoniou
  - Media:Antoniou--transactional_device_tree_and_overlays.pdf
- "devicetree: Kernel Internals and Practical Troubleshooting", ELC Europe 2014 by Frank Rowand
  - Media:Rowand--devicetree_kernel_internals.pdf
- "Device Tree, the Disaster so Far", ELC Europe 2013 by Mark Rutland
  - Media:Rutland-presentation_3.pdf
  - YouTube video
- "Best Practices for Long Term Support and Security of the Device-Tree (DT)" ELC Europe 2013 by Alison Chaiken
  - Media:Chaiken-DT_ELCE_2013.pdf
- "Board file to Device Tree Migration" ELC Europe 2013 by Pantelis Antoniou
  - Media:ELCE2013_-_DT_War.pdf
- "ARM support in the Linux kernel", Presented at FOSDEM 2013 by Thomas Petazzoni
  - https://archive.fosdem.org/2013/schedule/event/arm_in_the_linux_kernel/attachments/slides/273/export/events/attachments/arm_in_the_linux_kernel/slides/273/arm_support_kernel.pdf
  - Has good material on how device tree is part of the overall ARM architecture refactoring, with some details on how it is used
- "Linux kernel: consolidation in the ARM architecture support" - Libre Software Meeting, 2013 by Thomas Petazzoni
  - http://free-electrons.com/pub/conferences/2012/lsm/arm-kernel-consolidation/arm-kernel-consolidation.pdf
- "Experiences With Device Tree Support Development For ARM-Based SOC's", Thomas P. Abraham, ELC 2012
  - Media:Experiences_With_Device_Tree_Support_Development_For_ARM-Based_SOC's.pdf
  - slides and videos for ELC 2012: http://free-electrons.com/blog/elc-2012-videos/
- "Device Tree Status Report", Grant Likely, ELC Europe 2011

    - Slides and videos for ELC Europe 2011: http://free-electrons.com/blog/elce-2011-videos/

Notes on various sub-systems that device-tree describes

- "Pin Control Subsystem – Building Pins and GPIO from the ground up"
    - Presented at Linaro Connect, 2013 by Linus Walleij
    - http://www.df.lth.se/~triad/papers/pincontrol.pdf

older stuff

There is documentation describing device tree support (with information current as of 2006) in the Linux kernel source tree at: Documentation/powerpc/booting-without-of.txt

- "Using the Device Tree to Describe Embedded Hardware" - Grant Likely, Embedded Linux Conference, 2008
    - http://www.celinux.org/elc08_presentations/glikely--device-tree.pdf
- "A Symphony of Flavours: Using the device tree to describe embedded hardware" - Grant Likely and Josh Boyer - paper for OLS 2008
    - http://ols.fedoraproject.org/OLS/Reprints-2008/likely2-reprint.pdf
- Note from Device Tree Birds of a Feature session at OLS 2008:
    - http://lists.ozlabs.org/pipermail/devicetree-discuss/2008-July/000004.html
- Links to the Open Firmware device tree bindings and recommended practices which also apply to the FDT:
    - http://www.openfirmware.info/Bindings
- A view from outside from the FreeBSD ARM community:
    - http://wiki.freebsd.org/FreeBSDArmBoards

Tools

- Device Tree Compiler (dtc) - converts between the human editable device tree source "dts" format and the compact device tree blob "dtb" representation usable by the kernel or assembler source. dtc is also a dtb decompiler.
    - The linux version of dtc is maintained in `scripts/dtc/` in the kernel source directory.
    - The upstream project is maintained in
        - https://git.kernel.org/cgit/utils/dtc/dtc.git
        - git clone git://git.kernel.org/pub/scm/utils/dtc/dtc.git
- Xilinx EDK device-tree generator - Generates an FDT from Xilinx FPGA design files.
    - http://xilinx.wikidot.com/device-tree-generator

```
"The device tree generator is a Xilinx EDK tool that plugs into the
Automatic BSP Generation features of the tool, XPS"
```

Debugging

You can set CONFIG_PROC_DEVICETREE to be able to see the device tree information in /proc after booting. Build the kernel with this option set to 'Y', boot the kernel, then 'cd /proc/device-tree'

For newer kernels where the CONFIG_PROC_DEVICETREE option does not exist, /proc/device-tree will be created if CONFIG_PROC_FS is set to 'Y'.

You might also try CONFIG_DEBUG_DRIVER=Y.

Also, often, you can set the line: "#define DEBUG 1" to an individual C file, to produce add debug statements to the routines in that file. This will activate any pr_debug() lines in the source for that file.

Alternatively, you can add the following to drivers/of/Makefile:

```
CFLAGS_base.o := -DDEBUG
CFLAGS_device.o := -DDEBUG
CFLAGS_platform.o := -DDEBUG
CFLAGS_fdt.o := -DDEBUG
```

Device-Tree irc

The Device Tree irc channel is #devicetree on freenode.net.

Device-tree Mailing List

After July 2013:

```
http://vger.kernel.org/vger-lists.html#devicetree
archive: http://www.spinics.net/lists/devicetree/
```

Up through July 2013:

```
https://lists.ozlabs.org/listinfo/devicetree-discuss
archive: http://news.gmane.org/gmane.linux.drivers.devicetree
```

Categories:

- Bootloader
- Device tree
- Kernel

> From: eLinux.org

Device Tree frowand

Device Tree stuff from Frank Rowand

Resources for "Solving Device Tree Issues" talk

LinuxCon Japan - June 4, 2015

- PDF slides
- patch to make dtdiff work for .dts files in the Linux kernel source tree and in single file mode (v 150531_0043)
- patch to create dt_stat and dt_node_info scripts (~~v 150527_1902~~ v 150707_1852)
- notes on how to upload FDT and EDT from a target over a serial console (150531_0029)

Category:

- Device tree

> From: eLinux.org

Device tree future

Contents

Where Device Tree is Headed

Resources for the Linux Plumbers 2015 Device Tree Track

THIS SECTION IS UNDER CONSTRUCTION

Material to review before the event

The purpose of the Linux Plumbers conference is to **discuss** things. The conference is not a good place to go if you want to look at slides and listen to canned presentations.

The discussions will work better if the attendees have prepared in advance, and have a basic understanding of the technology and issues to be discussed. The goal of this section is to provide the resources needed to be prepared to discuss.

Device Tree 101

If you are new to Device Tree, these resources will start you on the path to a basic understanding.

- **An introduction**
  - Device trees I: Are we having fun yet? - Neil Brown, LWN.net November 2013
  - Device trees II: The harder parts - Neil Brown, LWN.net November 2013
  - "Device Tree for Dummies", ELC 2014 by Thomas Petazzoni
    - PDF
    - YouTube video
- **More advanced material**
  - "The Device Tree as a Stable ABI: A Fairy Tale?", ELC 2015 by Thomas Petazzoni
    - http://elinux.org/images/0/0a/The_Device_Tree_as_a_Stable_ABI-_A_Fairy_Tale%3F.pdf
  - "Device Tree, the Disaster so Far", ELC Europe 2013 by Mark Rutland
    - Media:Rutland-presentation_3.pdf
    - YouTube video

Overlays

- "Transactional Device Tree & Overlays: Making Reconfigurable Hardware Work", ELC 2015 by Pantelis Antoniou
  - PDF

    - YouTube video
- Problem statements
    - IO boards, eg beaglebone capes
    - PPC sub-tree beneath hot-plug PCI
        - http://www.spinics.net/lists/linux-pci/msg40740.html
        - It might be possible to use existing dynamic add and remove functions (CONFIG_OF_DYNAMIC) for this purpose
    - Quirks - TODO
        - [PATCH 0/4 Device Tree Quirks & the Beaglebone]
        - cpu card plugged into multiple carrier card variants, post manufacturing
    - devices present only during manufacturing
        - Dealing with optional i2c devices in a devicetree

documentation

dtc

Probe Ordering

- Tomeu Vizoso
    - First approach: [PATCH 00/21 On-demand device registration]
    - Second approach: [PATCH 00/13 Discover and probe dependencies]
- Other....

device tree debugging tools

Current draft of the schedule

Expect this to evolve.

```
01 -- Device Tree Overlays - Pantelis
      Devicetree overlay use in Juniper products - Guenter
02 -- folded into 01
03 -- Overlays, some times a good idea sometimes not. - Pantelis
04 -- Device Tree Documentation - Frank, Matt
05 -- Chat With The dtc Maintainers - Frank, the maintainers
06 -- Overlays and tools for sanity. - Pantelis
07 -- Device Tree Tools - Frank
08 -- Device Tree and parallel device probing - Pantelis
09 -- Device tree round up - Frank


     session  start
     length   offset
     ------   ------

  01    30       0
  02                  (folded into 01)
  03    15      :30
  04    15      :45

break   10     1:00

  05    30     1:10
  06    10     1:40
  07    15     1:55
  08    15     2:10
  09    10     2:20

      -----
       2:30
```

```
01 -- Device Tree Overlays - Pantelis
   Device Tree Overlays are now in the mainline kernel. This session
   will cover what they are, how they are used.

   As part of this session I will examine device tree overlays, device
   tree changeset, the phandle resolution mechanism, overlay overlap
   removal checks and finally device tree variants (or quirks).

   Devicetree overlay use in Juniper products - Guenter

   The Juniper use case will be discussed:

   At Juniper, we use devicetree overlays to manage a variety of cards
   which can be inserted and removed at runtime.

   In this session, I will describe the basic system architecture, our
   requirements, and why we decided to use devicetree overlays to meet
   those requirements. I will also dive into the actual implementation
   of our card management framework in the Linux kernel, and explore
   some of the limitations of the current devicetree overlay code.

02 -- was folded into 01

03 -- Overlays, some times a good idea sometimes not. - Pantelis
   This session will cover supported and not supported overlay cases.

04 -- Device Tree Documentation - Frank
   What device tree documentation and tutorials exist and where to find
   them. What is needed?

   What new documentation is expected this year?

   Can we bring consistency to the documentation style/syntax?

05 -- Chat With The dtc Maintainers - Frank
   This session is an opportunity to ask questions of the dtc maintainers
   or listen to their thoughts on dtc related topics.

06 -- Overlays and tools for sanity. - Pantelis
   Device Tree overlays represent a big change for the device tree in
   the kernel. Where as of old the device tree was something static,
   now it's something that can change at runtime.

   We could use some new tools to help us when creating them (compile
   time) and some kernel tooling to help when applying them (run time).

07 -- Device Tree Tools - Frank
   What tools exist to support device tree development and
   debugging? Where are they? What new tools have been proposed or
   requested?

08 -- Device Tree probe order and parallel device probing - Pantelis
   The new dynamic device tree capabilities entails marking not only
   the location of phandles but the references made to them. We can use
   that information to construct a device probe order schedule that can
   be used to support parallel device probing which is an obvious win
   for kernel boot time.

   If earlier sessions run long, this one may be shortened or deleted.

09 -- Device tree round up - Frank
   Review previous sessions, round up loose ends
```

Category:

- Device tree

> From: eLinux.org

Device tree history

Mailing list discussion

"Recent" (2009) discussion of "Flattened Device Tree" work on linux-embedded mailing list:

http://www.mail-archive.com/linux-embedded@vger.kernel.org/msg01721.html

Russell King is against adding support for FDT to the ARM platform (see whole thread for interesting discussion):

http://lkml.indiana.edu/hypermail/linux/kernel/0905.3/01942.html

But maybe Russell can be convinced:

http://lkml.indiana.edu/hypermail/linux/kernel/0905.3/03618.html

David Gibson defends FDT:

http://lkml.indiana.edu/hypermail/linux/kernel/0905.3/02304.html

The bindings review fire hose is clogged

The device tree bindings maintainership was broken apart from device tree maintainership on July 19, 2013, by commit f882820556af33b5aee5b9f0ba459620a9ab1c22 that created the "OPEN FIRMWARE AND FLATTENED DEVICE TREE BINDINGS" entry in the MAINTAINERS file.

```
    MAINTAINERS: Refactor device tree maintainership

    Device tree bindings require a lot more attention than they used to.
    We've got a group of volunteers willing to take over maintaining
    bindings. This patch adds them to the MAINTAINERS file.
```

Discussion at the October 2013 Kernel Summit in Edinburgh led to the creation of the kernel file Documentation/devicetree/bindings/submitting-patches.txt, including the following note:

```
   II. For kernel maintainers

     1) If you aren't comfortable reviewing a given binding, reply to it and ask
        the devicetree maintainers for guidance.  This will help them prioritize
        which ones to review and which ones are ok to let go.

     2) For driver (not subsystem) bindings: If you are comfortable with the
        binding, and it hasn't received an Acked-by from the devicetree
        maintainers after a few weeks, go ahead and take it.

      Subsystem bindings (anything affecting more than a single device)
      then getting a devicetree maintainer to review it is required.
```

Category:

- Device tree

> From: eLinux.org

Linux Drivers Device Tree Guide

Contents

Support of different hardware versions in a single driver

Examples of drivers that match more than one compatible string.

This list is not an endorsement of any particular technique. It is instead a (partial) list of some existing code in the Linux kernel.

The examples are not meant to capture each method entirely; they are instead meant to illustrate the basic concept.

Hardware Version in struct of_device_id.data

The hardware version is used throughout the driver to choose alternate actions.

drivers/iommu/arm-smmu.c:

```
static const struct of_device_id arm_smmu_of_match[] = {
        { .compatible = "arm,smmu-v1", .data = (void *)ARM_SMMU_V1 },
        { .compatible = "arm,smmu-v2", .data = (void *)ARM_SMMU_V2 },
        { .compatible = "arm,mmu-400", .data = (void *)ARM_SMMU_V1 },
        { .compatible = "arm,mmu-401", .data = (void *)ARM_SMMU_V1 },
        { .compatible = "arm,mmu-500", .data = (void *)ARM_SMMU_V2 },
        { },
};
MODULE_DEVICE_TABLE(of, arm_smmu_of_match);

static int arm_smmu_device_dt_probe(struct platform_device *pdev)
{
        const struct of_device_id *of_id;

        of_id = of_match_node(arm_smmu_of_match, dev->of_node);
        smmu->version = (enum arm_smmu_arch_version)of_id->data;

        ...

        if (smmu->version > ARM_SMMU_V1) {
                ...
        }
}

static struct platform_driver arm_smmu_driver = {
       .driver = {
                .name           = "arm-smmu",
                .of_match_table = of_match_ptr(arm_smmu_of_match),
        },
        .probe  = arm_smmu_device_dt_probe,
        .remove = arm_smmu_device_remove,
};
```

Function Call Table pointer in struct of_device_id.data

The function call table is used throughout the driver to choose alternate actions.

drivers/iio/adc/xilinx-xadc-core.c:

```
static const struct xadc_ops xadc_zynq_ops = {
        .read = xadc_zynq_read_adc_reg,
        .write = xadc_zynq_write_adc_reg,
        .setup = xadc_zynq_setup,
        .get_dclk_rate = xadc_zynq_get_dclk_rate,
        .interrupt_handler = xadc_zynq_interrupt_handler,
        .threaded_interrupt_handler = xadc_zynq_threaded_interrupt_handler,
        .update_alarm = xadc_zynq_update_alarm,
};

static const struct of_device_id xadc_of_match_table[] = {
        { .compatible = "xlnx,zynq-xadc-1.00.a", (void *)&xadc_zynq_ops },
        { .compatible = "xlnx,axi-xadc-1.00.a", (void *)&xadc_axi_ops },
        { },
};
MODULE_DEVICE_TABLE(of, xadc_of_match_table);

static int xadc_probe(struct platform_device *pdev)
{
        const struct of_device_id *id;
        id = of_match_node(xadc_of_match_table, pdev->dev.of_node);
        ...
        indio_dev = devm_iio_device_alloc(&pdev->dev, sizeof(*xadc));
        xadc = iio_priv(indio_dev);
        ...
        xadc->ops = id->data;
        ...
        ret = xadc->ops->setup(pdev, indio_dev, irq);
        ...
}

static struct platform_driver xadc_driver = {
        .probe = xadc_probe,
        .remove = xadc_remove,
        .driver = {
                .name = "xadc",
                .of_match_table = xadc_of_match_table,
        },
};
module_platform_driver(xadc_driver);
```

Hardware Description pointer in struct of_device_id.data

The hardware description data is used to configure the device.

This struct pointed to by struct of_device_id.data in this example includes a function call table in addition to the hardware description fields.

drivers/iio/adc/twl6030-gpadc.c:

```
static const struct twl6030_gpadc_platform_data twl6030_pdata = {
        .iio_channels = twl6030_gpadc_iio_channels,
        .nchannels = TWL6030_GPADC_USED_CHANNELS,
        .ideal = twl6030_ideal,
        .start_conversion = twl6030_start_conversion,
        .channel_to_reg = twl6030_channel_to_reg,
        .calibrate = twl6030_calibration,
};

static const struct of_device_id of_twl6030_match_tbl[] = {
        {
                .compatible = "ti,twl6030-gpadc",
                .data = &twl6030_pdata,
        },
        {
                .compatible = "ti,twl6032-gpadc",
                .data = &twl6032_pdata,
        },
        { /* end */ }
};

static int twl6030_gpadc_probe(struct platform_device *pdev)
{
        const struct of_device_id *match;
        const struct twl6030_gpadc_platform_data *pdata;

        match = of_match_device(of_twl6030_match_tbl, dev);
        pdata = match->data;
        indio_dev = devm_iio_device_alloc(dev, sizeof(*gpadc));
        gpadc = iio_priv(indio_dev);
        gpadc->pdata = pdata;
        platform_set_drvdata(pdev, indio_dev);
        ...
        ret = pdata->calibrate(gpadc);
        ...
        indio_dev->channels = pdata->iio_channels;
        indio_dev->num_channels = pdata->nchannels;
}
```

Category:

- Device tree

> From: eLinux.org

Hardware Hacking

Information about running Linux on devices that are or have been available to the general public. This includes both officially supported devices and project devices (or devices that unofficially run Linux).

Contents

Project Devices

- Literati
- InnoTab
- Opensourcemid K7 MID OMAP3530 tablet
- CR48 Google Netbook
- Ben NanoNote
- TCube
- Mobile Pro
- LeapFrog Products
  - Didj
  - Leapster Explorer
  - Leappad Explorer
  - Leapster
- DCT 5000
- Pixter
- Pixter Multimedia
- TvNow
- ZipIt
- Zipit2 - the new Zipit with better hardware
- JuiceBox
- DHT-Walnut
- FX3002 watch from fossil.
- Hisense - USDTV HDTV Tuner DB-2010 running Linux
- enc28j60 - single chip 10baseT ethernet with SPI interface
- R8610_Based_WAP - tiny x86 compatible WAPs with internal 2.5" HD runs Linux
- SMC WSKP100 - A wifi Skype phone from SMC
- OpenTom - community project around GPS TomTom devices
- EBR-1000EP - Sony Librié
- VGF-CP1 - Sony VAIO Digital Picture Frame with WiFi
- Peek
- Reciva_Barracuda
- Wavefinder Software Defined Radio

- CT-PC89E - 8.9" netbook powered by S3C6410
- Programmers Hardware Database
  - various hardware information including hardware hacking projects
- DF3210 - Parrot bluetooth digital photo frame

Supported Devices

- Linksys NSLU2. See also http://www.nslu2-linux.org
- Linksys WRT54GL
- Neuros OSD
- Nokia 770
- Nokia 800 and Nokia 810
- Sharp Zaurus and NetWalker running Ubuntu
- AML Handheld devices
- Pandora handheld console

Automotive

- NaviEngine
- Automotive Communications
  - CAN Bus
  - BEAN Bus
  - AVC-LAN

Industrial

- Industrial Communications
  - CAN Bus

Miniclusters

- Portable cluster hardware and related technologies for parallel computer tutorials, demonstrations, research.

Here is a list of original projects.

- See http://eri.ca.sandia.gov/eri/related_technologies.html
- See http://www.linuxdevices.com/articles/AT2143783710.html
- See http://www.coreboot.org/data/clusters/bento/index.html
- See http://www.coreboot.org/data/clusters/dq/index.html
- See http://coreboot.org
- See http://www.youtube.com/watch?v=UPyn9krjIRc

Hardware Tools and Information

- Flameman & legacy's hardware hacking pages (68k, mips, powerpc, blackfin, avr, and much more)
- Flyswatter USB JTAG and RS-232 adapter
- Sparkfun_Camera - Low cost cmos camera
- Ez_Usb - 8051 based usb devices
- Mini_LA - open source logic analyzer

- NTSC_Bitbang - detailed concepts on NTSC
- Lithium_Ion_Charger
  - Li-Ion battery charger design
- Libertas SDIO - Marvell Libertas SDIO information
- Nand_Flash256 - common 256MB nand flash devices
- Nor_vs_Nand - data comparisons between Nor and Nand Flash
- TUSB2046B - four port usb chipset

Software Tools and Information

- SM501-User_Level_Device_Driver
- Board_Bringup_Utilities
- MUSB resources
- Hello_World_in_C

Category:

- Hardware Hacking

> From: eLinux.org

AML Products

About AML

AML (American Microsystems, Ltd.) was founded in 1983 to respond to a need in the bar code marketplace for high performance, easy to use, and cost-effective bar code and data collection products. Since 1983, AML and its partners have helped thousands of companies around the globe to increase business efficiency and productivity - in manufacturing, warehousing, retail, health care, finance, government, and education.

- http://www.amltd.com
- http://linuxdevices.com/articles/AT5552849613.html
- http://linuxdevices.com/articles/AT2525542505.html
- http://linuxdevices.com/articles/AT9973611275.html

Products

- M8050
- M5900

Category:

- Products

> From: eLinux.org

Automotive Communications

Modern Toyota automobiles typically have 3 major bus systems: the CAN bus, the BEAN bus, and the AVC-LAN bus. Each of the system buses must inter-operate with the other system buses, so we need a basic understanding of them. All of the bus systems consist of some “backbone” of a bus network and many individual nodes or Electronic Control Units (ECU) such as amplifiers, receivers, CD-changers, automatic locks, or any other component.

- CAN Bus
- BEAN Bus
- AVC-LAN

Category:

- Networking

From: eLinux.org

AVC-LAN

Contents

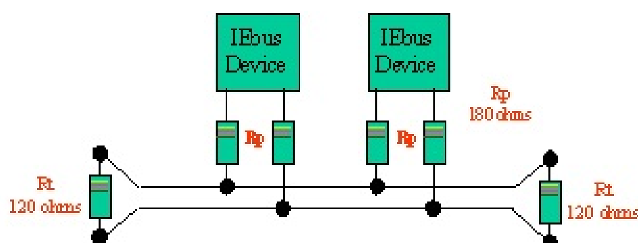
Introduction

The AVC-LAN is part of the Body Electrical System Control and manages all audio and video related functions. The audio bus is described in Toyota's own words as follows:

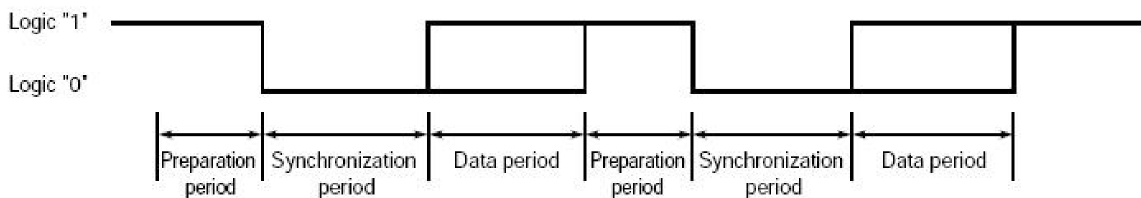
```
AVC-LAN consist of audio visual systems such as the multi-display, navigation ECU, radio and
player, stereo component amplifier and gateway ECU. Gateway ECU has communication
circuit to correspond with different types of communication data. Different types of
communication data can be shared among communication parts after it goes through gateway
ECU (System Circuits: AVC-LAN Bus, “Prius Wiring Diagram” 80).
```

The system itself can be considered a subset of the IEBus Standard, which was delevoped by NEC electronics for automotive use. It is slightly faster than the BEAN Bus and has more data per command, but is still much slower than CAN Bus, since realtime speeds are not required.

Connections



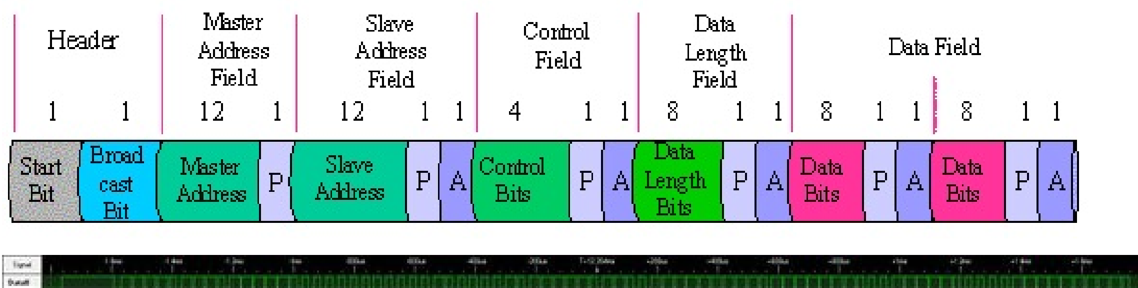
Waveform



Logic 1: The potential difference between the bus lines (the BUS+ and BUS- pins) is 20 mV or less (low level).
Logic 0: The potential difference between the bus lines (the BUS+ and BUS- pins) is 120 mV or more (high level).

- Prep 7uS
- Sync 20uS
- Data 13uS

Protocol



This is an AVC-LAN message captured using a logic probe.

Hardware

Specific

- Renesas HA12240 Differential Transceiver
  - A2240 Package Marking
  - CA0008 Package Marking
  - CA0013 Package Marking
- NEC uPD6708 IEBus (Inter Equipment Bus) Protocol Controller
- NEC uPD72042 IEBus (Inter Equipment Bus) Protocol Controller

General Purpose

- LM239 comparator (can be used for receiver)
- PCA82C250 CAN Bus (can be used for driver)
- MCP2551 CAN Bus (can be used for driver)
- RS-485 Differential Transceiver (can be used for both receiver and driver)
  - SN65HVD11 +3.3V
  - SN75HVD11 +5v

Hacks

- Marcin Slonicki's Toyota Corolla AVC-Lan MP3 player using:

  - ATmega8 is clocked at 14.7456 MHz with an external crystal
  - PCA82C250 CAN bus driver is used to drive the IEBus
  - LM239 comparator is required on top of the PCA82C250 to read from the IEBus
- Louis Frigon's Toyota Auxiliary Audio Input Enabler using:
  - ATmega8 is clocked at 8.00 MHz with the internal RC
  - ATmega8's internal analog comparator handles the receive side
  - ATmega8's two GPIOs as antagonist output pins perform the bus driver

Documentation Credit

- Jeremiah J. Flerchinger Whitepaper

Category:

- Networking

> From: eLinux.org

BEAN Bus

40x40px	This article **is incomplete**. Please help to improve the section, or discuss the issue on the talk page.

BEAN is a Toyota specific standard that operates in a daisy chain loop. It networks the body electric system and manages items like air conditioning and meter displays. This system is not as time critical as the CAN network and operates over a single voltage driven wire at 10kbps with 1-11 Bytes of data per transmission command.

(Jeremiah J. Flerchinger Source)

Categories:

- Pages with broken file links
- Articles to be expanded
- Networking

> From: eLinux.org

Board Bringup Utilities

Description	Tool
Framebuffer	fb-test-app git repo (original fb-test)
Input Events	evtest
Physical Memory	devmem2
I2C Interface	I2C Tools
OMAP Booting	OMAP U-boot Utils
OMAP4 USB Booting	OMAP4 USB Booting
Uart	Uart Loopback
OMAP overlay swapper	overlayswap
OMAP memory speed test	memspeed
OMAP4 EMIF cycle speed	omap4_emif
OMAP4 EMIF performance	omap4_ddrstat

```
To compile single .c source files such as fb-test.c,
use compiler command line:

# arm-none-linux-gnueabi-gcc fbtest.c -o fb-test

this will generate the binary fb-test for use on your
system where arm-none-linux-gnueabi- is the cross compiler
that you have installed on your host system.
```

- minimal kernel config for 4430sdp
- minimal rootfs for OMAP3/OMAP4
- test kernel for PandaBoard

Category:

- Development Tools

> From: eLinux.org

CAN Bus

Contents

Overview

The CAN bus is an ISO standard bus originally developed for vehicles. It manages the Chassis Electrical System Control and is responsible for critical activities like engine electrical, and skid control. This system is also used to provide vehicle diagnostic information for maintenance. A multi-star configuration seems typical of this bus with a primary bus line that branches into sub bus lines at its extremities then attaches to multiple device nodes. Differential voltage is applied over twisted pair at 1.5 to 2.5V and 2.5 to 3.5V for noise resistant signaling. Bit rates up to 1 Mbit/s are possible at network lengths below 40 m. Decreasing the bit rate allows longer network distances (e.g., 500 m at 125 kbit/s). (Jeremiah J. Flerchinger Source) Controllers supporting CAN FD, an enhanced CAN version with frames up to 64 byte and bit rates up to 4 Mbit/s, will be available in the second half of 2014. A can4linux version supportig CAN FD on a IFI CAN is ready to be used.

Although developed as car communication network CAN is used in many other areas, industrial, medical, maritime laboratory and more. Most often with a CAN based higher layer protocol like CANopen on top of it.

Additional information can be found at:

http://en.wikipedia.org/wiki/CAN_bus

SocketCAN News

CiA CAN in Automation CAN user association

CAN Wiki

CAN FD Specification Version 1.0

CAN Support in Linux

CAN is supported by Linux device drivers. Mainly two types exist. Character device based drivers and network socket based drivers. The Linux kernel supports CAN with the SocketCAN framework.

- SocketCAN Documentation
- mailing list for Linux Kernel CAN development
- linux-can git repository
- linux-can-next git repository
- Berlios Project Page (obsolete)

One of the character based drivers is can4linux.

- SourceForge project page
- German Wikipedia article
- English Wikipedia article

SocketCAN Supported Protocols

- RAW: send & receive raw CAN frames
- BCM: Broadcast manager, offload repetitive work to the Linux kernel
- ISOTP ...
- SAE J1939

SocketCAN Supported Controllers

- Microchip MCP251x
- Atmel AT91 SoCs
- ESD 331 CAN Cards
- NXP (Philips) SJA1000
- Freescale MPC52xx SoCs
- Bosch CC770
- Intel AN82527
- TIs SoCs
- Serial/network devices utilizing ASCII protocol (slcan driver)

Vendor	Device Name	Driver Module Name	Controller	Kconfig Option	Linux Mainline	R
VScom	NET-CAN	slcan	(unknown)	CONFIG_CAN_SLCAN	2.6.38	ne
VScom	PCI-2CAN	slcan	(unknown)	CONFIG_CAN_SLCAN	2.6.38	ne
VScom	SER-CAN	slcan	(unknown)	CONFIG_CAN_SLCAN	2.6.38	ne
VScom	USB-CAN	slcan	(unknown)	CONFIG_CAN_SLCAN	2.6.38	ne
LAWICEL	CAN232	slcan	(unknown)	CONFIG_CAN_SLCAN	2.6.38	ne
LAWICEL	CANUSB	slcan	(unknown)	CONFIG_CAN_SLCAN	2.6.38	ne
PEAK	PCAN-PCI	peak_pci	sja1000	CONFIG_CAN_PEAK_PCI	3.2	su [n va
PEAK	PCAN-USB	peak_usb	(unknown)	CONFIG_CAN_PEAK_USB	3.4	su U su
Kvaser	PCIcanx	kvaser_pci	sja1000	CONFIG_CAN_KVASER_PCI	2.6.31	su [n va
Kvaser	Leaf	kvaser_usb	(unknown)	CONFIG_CAN_KVASER_USB	3.8	su M R de kv
Kvaser	USBCan-II	kvaser_usb	(unknown)	CONFIG_CAN_KVASER_USB	commits pulled-in for the 3.20 release	su
EMS Wünsche	CPC-Card	ems_pcmcia	sja1000	CONFIG_CAN_EMS_PCMCIA	3.2	di
EMS Wünsche	CPC-USB/ARM7	ems_usb	(unknown)	CONFIG_CAN_EMS_USB	3.2	
EMS Wünsche	CPC-PCI/PCIe	ems_pci	sja1000	CONFIG_CAN_EMS_PCI	3.2	up
EMS Wünsche	CPC-PC104P	ems_pci	sja1000	CONFIG_CAN_EMS_PCI	3.2	up P
8devices	USB2CAN	usb_8dev	(STR750FV2)	CONFIG_CAN_8DEV_USB	3.9	S
Softing	CANcard2	softing_cs	sja1000 or NEC-005(?)	CONFIG_CAN_SOFTING_CS	2.6.38	S {C P N fw

SocketCAN Support in Programming Languages/Environments

- Android
- Java
- Python
- Python library for CAN
- Matlab/Simulink blocks to send and receive CAN messages

can4linux Supported Controllers

- Allwinner A20 with integrated CAN (on the popular BananaPi single-board computer.)
- Analog Devices BlackFin BF537
- Atmel AT91 SoCs
- Freescale FlexCAN (ColdFire 5282, i.MX25, i.MX28, i.MX35)
- Intel 82527 (the replacement Bosch CC770 should work)
- Microchip Stand Alone CAN MCP2515
- NXP Stand Alone CAN SJA1000 (on different ISA or PCI/PCIe boards)
- Xilinx Zynq with XCAN
- 'virtual' CAN mode without CAN hardware
- 'virtual' CAN mode supporting CAN FD
- IFI CAN FPGA IP, in classic CAN mode and CAN FD mode

can4linux Support in Programming Languages/Environments

- C - many examples and useful applications are provided with the package, check can4linux-examples/
- Tcl/Tk also in can4linux-examples/
- Python also in can4linux-examples/

CAN Controllers Emulation (WIP/experimental)

- SJA1000 CAN controller based PCI board emulation for QEMU
  - Cards models provided:
    - Simple memory mapped SJA1000 in the first PCI BAR with ad-hoc PCI ID
    - Kvaser PCIcan-S single I/O mapped SJA1000 model compatible with kvaser_pci Linux driver on guest side
  - The emulated CAN buses can be connected to virtual or physical SocketCAN interface if Linux is used as host system
  - Project repository: https://github.com/CTU-IIG/qemu
  - Work started by 2013 GSoC project when RTEMS project donated its slot to work on QEMU CAN support - see RTEMS related page for more info and use instructions

SocketCAN Bechmarking

- CAN gateway timing analysis and repository with benchmark infrastructure

SocketCAN Tutorials

- Bringing CAN interface up
- can-utils
- libsocketcan
- libnl

Category:

- Networking

> From: eLinux.org

CT-PC89E

Contents

Chitech CT-PC89E

- Current status: GPL Violation by Chitech. Communication has ceased by Chitech. Chitech have not responded to or acknowledged communications since 20th March 2010.
- Current kernel reverse-engineering status: partial (useable). Screen, USB, Keyboard, Mouse, Ethernet are all working (proviso: with the DS2431 EEPROM storing the MAC code and no driver for reading it yet, a random MAC is currently created on each boot)
- Current Debian-Installer status: success! completed!
- Current u-boot reverse-engineering status: not started yet. hardware "switch" discovered which makes booting from external SDcard possible. u-boot reverse-engineering can now begin (without risk of bricking)

The CT-PC89E is a low-cost netbook with an 8.9in 1024x600 screen, weighing only 720 grammes (0.72kg). In size it's *approximately* 23.5 x 16.5 x 2.5 cm (9.25 x 6.5 x 1 in). Its 667mhz Samsung S3C6410 embedded ARM CPU is on a factory-upgradeable SO-DIMM which also has, in the standard low-cost option, 256mb of RAM and 2gb of NAND Flash. The rest of the features are pretty much "standard" fare for a low-cost netbook: 2x USB2, stereo speakers, microphone, SD-Card slot, headphone and microphone sockets, and 802.11 WIFI. A low-cost (0.3mp) built-in Webcam is available as an option ($USD 2 for a 20,000 units order). To further save on cost, there is a micro VGA output, but by default the IC to enable it is again optional (again, $USD 2 for a 20,000 order). Also, the design has two internal USB2-capable (only) PCI-express slots, which can take 50x30mm PCI-e cards. One is occupied with the RALink RT2070 WIFI, whilst the other is designed to take a 3G or an EDGE modem: there is even a slot for a SIM card (next to the SD card slot).

As this machine is very new, only a few brave Debian-ARM souls have bought it so far, direct from the factory in China, in order to evaluate it and help re-engineer it. We're aware that one other U.S. customer has ordered a batch of them, thus guaranteeing its production over the next few months (as of Feb 2010). The nice surprise is that far from being truly dreadful, the embedded OS on the device, from http://mid-fun.com is actually pretty good: it's called MOS and the web site is here: http://mid-linux.org. As of yet, we've been unable to reach Mid-Fun to get them to provide the root password and the GPL source code of the OS, but that's okay because we've discovered three security flaws in two days, each of which gives full root access to the machine. (24feb2010: by running "john" on the DES64/64 root passwd entry, we've established that the root password is mos2010)

So, at this early stage, for more information please contact mailto:lkcl@lkcl.net, mailto:luke.leighton@gmail.com or Asia Sourcing http://www.asiasourcing.net/ and as soon as we find or hear from a retail outlet or a distributor brave enough to sell these systems we'll let you know.

Also worth noting: we're currently asking the factory for a price on engineering an SO-DIMM with 512mb of DDR2 RAM and an 833mhz Samsung ARM Cortex A8: the S5PC100, which is the same CPU as used in the iPhone 3G. This would ironically not only reduce the price of the system, because DDR1 RAM is actually more expensive than DDR2, but also give it a huge performance jump, without increasing power consumption (the S5PC100 is a 45nm part and the S3C6410 is 65nm).

The mailing list is presently being kindly hosted by Alain Williams, at http://lists.phcomp.co.uk/mailman/listinfo/arm-netbook

There is a description of the system, and photos (internal and external), here: http://lkcl.net/arm_systems/CT-PC89E

Hardware Summary

Netbook Specifications

Display	8.9" 1024x600 LCD
Weight	720g (0.72kg)
Dimensions	23.5 x 16.5 x 2.5 cm (9.25 x 6.5 x 1 in) *(aprox.)*
Battery	7.4V 15.5wh 2100mAh
AC Adapter	9V 2A

Main board

Audio Controller	AC97 (*Wolfson WM9715G*)
Ethernet Controller	Davicom DM9000DEP dongle required
VGA Controller	Chrontel 7026B-TF *(optional)* dongle required

200-pin SO-DIMM
(*seatron-cpu-v1.1*)

SoC	Samsung ARM11 S3C6410 667mhz (*s3c6410XH-66*)
NAND	SanDisk 2GB (*SDIN2C2-2G*)
RAM	256M DDR1 (2 x Hynix H5MS1G62MFP)
Boot Switch	boot from NAND or SD *(optional)* *(outer means NAND, inner means SD)*

Info Pages

- CT-PC89E_Debian_Installer
  - how to install debian on the CT-PC89E using Debian-Installer.
- CT-PC89E_Debian - how to install debian "hack-style" by dropping a pre-build root filesystem onto the CT-PC89E.
- CT-PC89E_Debian_Installer_Building
  - how to build debian-installer for the CT-PC89E (only needed by developers)
- CT-PC89E_Bugs

Technical Info

This section contains technical info needed to configure a linux kernel. Taken from kern.log

- LCD Type is: N10116 (innolux?) this is a lie, it's actually a B089AW01 V.1
- DM9000 is at f7600300,f7600304 IRQ74
- fb0 map video memory: ff200000:0012c000 dma=5e200000
- fb1 map video memory: ff32d000:0012c000 dma=5e400000
- fb0 map video memory: ff45a000:0012c000 dma=5e600000
- mmc0 (NAND flash) address 9535
- mmc1 (SDcard) address a95c
- audio: WM9713/WM9714

From this it's been possible to establish that the kernel version is similar to the SmartQ5 MID device, version 2.6.24.7. current status as of 12mar2010 is that boot has been achieved, using the above information, with DM9000 and USB host still not yet correctly initialised.

LCD Data

- B089AW01 V.1 http://www.asdatech.com/modules/classifieds/datasheet/B089AW01%20V.1.pdf

/proc/iomem

```
debianarmel:~# cat /proc/iomem
50000000-5e9fffff : System RAM
  50025000-503d1fff : Kernel text
  503d2000-5041daa5 : Kernel data
70200000-702fffff : s3c2410-nand
74300000-743fffff : s3c2410-ohci
  74300000-743fffff : ohci_hcd
76100000-761fffff : s3c-g2d
77000000-770fffff : s3c-vpp
77100000-771fffff : s3c2410-lcd
  77100000-771fffff : s3c-lcd
78800000-78bfffff : s3c-jpeg
7c200000-7c2fffff : s3c-hsmmc0.0
  7c200000-7c2fffff : s3c-hsmmc0
7c300000-7c3fffff : s3c-hsmmc1.1
  7c300000-7c3fffff : s3c-hsmmc1
7e002000-7e002fff : s3c-mfc
7e005000-7e0050ff : s3c2410-rtc
  7e005000-7e0050ff : s3c2410-rtc
7e00b000-7e00bfff : s3c-adc
7f001000-7f001fff : s3c-ac97
7f004000-7f004fff : s3c2410-i2c
  7f004000-7f004fff : s3c2410-i2c
7f005000-7f0053ff : s3c-uart.0
  7f005000-7f0050ff : s3c-uart
7f005400-7f0057ff : s3c-uart.1
  7f005400-7f0054ff : s3c-uart
7f005800-7f005bff : s3c-uart.2
  7f005800-7f0058ff : s3c-uart
7f005c00-7f005fff : s3c-uart.3
  7f005c00-7f005cff : s3c-uart
7f008000-7f008fff : s3c-ds2431
f7600300-f76fffff : dm9000.0
  f7600300-f7600303 : eth0
```

Kernel Hacking Progress

We're presently negotiating with Chitech, who have officially declined to release kernel source code, and all other source code, on the usual basis that they want their money back from the investment so far, in blatant disregard of the GPL and LGPL licenses. Thus it is necessary to reconsider Chitech until such time as they comply, and thus it is necessary to reverse-engineer the linux kernel and the boot process.

kernel

fjp and lkcl have been working to adapt the mer-smartq 2.6.24.7 kernel, which is an amalgam of 2.6.24, samsung's patch and various modifications hard-coded to support the smartq 4in and 7in LCD screens. lkcl has a legally-licensed copy of IDApro and has been using that to reverse-engineer the kernel functions, taking the addresses from /proc/kallsyms on a running machine and matching them to an uncompressed kernel.

So far we have been able to get the 1024x600 LCD panel up and running, and have hacked kernel/printk.c to modify the console line to "tty1" in order to bypass the inability to use ttySAC0 serial console, for debug purposes.

The key hardware that is missing which would make a useful kernel is the GL850G USB hubs. It has not yet been possible to establish how those are fired up. Close examination of the PCB shows that there is nothing connected to Pin 13 (RESET); the next best guess is that there are some GPIOs which are required to be pulled HIGH or LOW, in order to power the GL850Gs.

The Ethernet DM9000 appears to be identical to the standard samsung developer kit: the default parameters worked immediately. However, the MAC address is stored in a DM2431. We have found a patch to the w1 driver which adds support for DM2431, however have not yet used it. Further investigation shows that the SO-DIMM, designed by Seatron, has the DM2431 connected to GPIO Bank N pins, and thus the driver performs bit-level reads and writes using GPIO pullups followed by sleep commands of specific length. There are signs of some extremely weird functions in the disassembly listing, with SHA-1 being used and functions with the word "Secret" in them are not a particularly good sign. It looks like the DM2431 is being used to store keys which are believed to somehow magically be secure.

__apm_get_power_status

this function has been hacked to read from the S3C ADC. GPIO bank N pin 2 is set to cfgpin of 0 and pullup of 0, following which a function GetAdcValue is called. it is therefore quite likely that the ADC input is multiplexed for multiple purposes. there appears to be no sign of mutex locking or configuration around the GPIO bank N pin setting, thus making it entirely likely that corrupted values will be read due to race conditions.

s3cfb_set_brightness

the s3cfb_set_brightness function uses a PWM timer, s3c6410_timer_setup channel 0, which uses GPIO Bank F, Pin 14, as a PWM. this timer is botch-reset in the arch_reset() function, which has been hacked about and _should_ be using the watchdog timer, but is not.

arch_reset

arch_reset, in include/asm-arm/arch-s3c2410/system.h, should be used to do a watchdog reset. instead, it has been hacked about to reset GPIO Bank H Pin 8 and GPIO Bank F Pin 14 (the PWM for the backlight).

initrd

So far we have been able to establish that the standard bootpImage boot process works absolutely fine. This has been established by

- unpacking the zImage_dt_update with dd if=zImage_dt_update of=vmlinuz_zimage_update bs=1 skip=13428 followed by zcat
- extracting the initrd by searching for a 2nd zcat signature, which is at offset 78848
- noting that the initrd is a compressed CPIO filesystem
- adding initrd_phys-$(CONFIG_MACH_CTPC89E) := 0x50800000 to arch/arm/mach-s3c2410/Makefile.boot

- compiling with make bootpImage with INITRD=\

This last step recreates the exact same process used by Chitech to create the zImage_gt_update. The next step is to create an alternative initrd, such that e.g. Debian-Installer can be packed into it.

Other Brand Names

The CT-PC89E is beginning to appear under different brand names.

- HJ-8901 [1]
  - this page states that the machine is suitable for WINCE, which is not supplied (the page used to state that the OSes Linux and Lunix were preinstalled).

See also

- CT-PC89E/Emdebian

External links

- http://www.youtube.com/watch?v=3_SIGTNj4vY
- http://www.youtube.com/watch?v=RzzUj8VM1nE

Category:

- CT-PC89E

> From: eLinux.org

DCT 5000

- http://www.gi.com/noflash/digicable_dct5000.html
- http://gicout60.gic.gi.com/customer_docs/
- http://www.linuxdevices.com/news/NS3134551333.html

note these are _not_ the 5200. no HD/PVR functionality in the base model

- cable in
- RF out
- conv in
- conv out
- 2 ieee-1394
- r-l-v in and out
- 1 rs-232
- vga out
- parallel
- ethernet PCNet
- 2 usb.
- data hs
- ir connector
- SPDIF
- S-Video
- smart card
- pcmcia slot. The card located under the mainboard has a legend on it: "DCT5000 PCMCIA Conn".

Includes US power cord, and 8" RF cable, 22 page hookup manual.

- nec vrc5476-83e North Bridge
- nec vr5432 CPU
- Sony CXD3204 IEEE 1394
- Sony CXA2094
- AMD 79C970AKC Ethernet
- 2) InTel E28F320J5-120 flash
- 1) InTel TE28F320C3 flash
- 1) AMD 29DL323CB flash
- Crystal Semi CS4620 EOL :(
- Crystal Semi CS4298 Programmer's manual EOL
- ATI Rage XL
- ATI Rage Theater
- two RF modules
- Broadcom BCM3120 Set-Top Box Tranceiver
- Broadcom BCM3300 Single Chip DOCSIS Cable Modem
- Broadcom 7010 SetTop Box Decoder
- 4) MitsubishiM2V64S30BTP RAM
- 44 pin header. (ide we presume)
- 144 pin SO-DIMM slot. (ram?)
- ALI M1543 Southbridge/SuperI/O
- EPCOS M3654K - 45.75MHz IF splitter ***(front board F202)***
- Altera 7128S CPLD ***(mainboard U41)***
- Lattice M4A3-32/32-10VC CPLD (***mainboard U42 & U37***)
- Macronix MX23L1611 Mask ROM

    - Possible Flash Equivalent
- Winbond 32 x 8 SRAM

The processor info can be found here

Or direct links:

- Datasheet
- Users manual vol. 1
- Users manual vol. 2
- Instruction set for MIPS IV ISA

Ken McGuire got his on 3/15/2003 and found the following:

- the serial port operates at 38400 baud.
- Hold down the menu and info button, keep holding them while plugging in the box, "boot" will appear (after about 5 seconds of holding) on the display and some stuff will come out the serial port.
- the cursor and channel buttons will cycle through a menu: cold, hunt, disp, n dl, run.
- pressing select at each ot the menu items will do something, and produce data on the RS-232 port.
- there is a reset button inside that does the same as powering on. labelled SW900.
- holding just the menu button during reset gets the "boot" display without any serial port activity.
- the "pcmcia slot" connects to the motherboard via a 24 pin connector J402, so it's probably not true, or complete PCMCIA.
- there is no video out of any of the outputs, VGA, S-VID, video out.
- JTAG comes out to J7, a 10 pin header with half the pins grounded.
- pin-out at: JTAG pin-out
- J7 with connector (it's a .050" spacing): J7conn picture

Here is what comes out the serial port at boot time:

- DCT5000 Boot ROM v1.17
- Copyright (c) 1999 by General Instrument Corp.

- SUDB 0xBF000004

- 58 00 00 00 53 55 44 42 00 00 11 01 07 00 00 00
- E9 E2 EB E2 01 BC 20 02 74 7B 0B 00 03 10 20 20
- 20 20 20 20 20 20 30 30 30 2E 00 30 00 00 00 00
- 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 BF
- 00 00 00 00 00 00 00 00 00 00 00 00 00 06 00 BF
- 00 00 00 00 03 01 CF CA

T0mW found that while holding the INFO button down while powering up, or RESET, it will give a verbose debug of the frequencies it is "hunting" on.

MonMotha got his DCT5000 on 3/19/2003. I haven't done much with it other than take the top off, but I'll update this page as I go. So far I'm just attempting to find some of the missing doc links.

sorphin got both of his DCT-5000's on 3/18/2003, and is doing some digging, also found this link [1] to a page where a guy is working on mapping out how the DCT-2000 and 5000 work.

Here's a list of test points, add to them as they are discovered:

- TP9 (next to the CPU, towards the back): Pin 76 of CPU - Sys Clock
- TP11 (Next to U33): SDRAM Clock
- TP510: Ground

Notes TimRiker "found":

Connect a straight-through serial cable from the COM1 port on the host computer to the RS-232 port on the target system. Make sure that the RS-232 connector is attached to internal connector labelled J800. 9600-8-n-1

Power ON the target board. When it boots, a # prompt appears.

Enter the following commands at the # prompt:

```
export BOOTFILE=vmlinus
export BOOTHOST=
export NETADDR=
export NETMASK=
export BROADCAST=
export ETHERHWADDR=
```

TimRiker is not sure if this is after installing a different ROM or something... Still looking.

Ken McGuire searches through the binary images of Tim's ROM and the original ROM and doesn't find any of the above strings. Nothing comes out his DCT5000's J800 on power-up either.

Here's the DOCSIS configuration file of the box:

```
NetworkAccess = Yes

Start ClassOfService
ClassID = 1
MaxRateDown = 1500000
MaxRateUp = 128000
PriorityUp = 1
GuaranteedUp = 1280
MaxBurstUp = 0
PrivacyEnable = Yes
EndOf ClassOfService

MaxCPE = 4
```

Category:

- Hardware Hacking

> From: eLinux.org

DHT-Walnut

other DHT-Walnut pages: Flameman/dht-walnut

Contents

Overview

DHT-Walnut is a shortened name for the Digital Home Technologies PCB 01070201 Rev. 1.1. The DHT-Walnut board is based on the AMCC-Walnut (still available, and still expensive :P). It is a more improved board described at www.amcc.com. A "walnut" searching will inform about schematic and software. Also keep attention @ DENX: they have developed a lot for the AMCC-Walnut board, and you could find a linux patch for your problem, prettier documentation, developer suit, and much more: in case, you are suggested to have a look to their web pages.

currently available HERE Oops, **sold out** 2/6/2006

The board consists of:

- PowerPC 405GP running at 266mhz
- PC133 SDRAM slot, currently, only supports single sidded DIMMs
- On-chip 405GP ethernet, board doesn't have an ethernet MAC address
- DCE serial port, speeds up to 230k, only tested to 115200
- two pci slots, keyed for 5V only cards
- Promise Technologies PDC20265 IDE
- 512k of boot flash AMD 29LV040B datasheet
- 12v dc powered
- PPC40XX JTAG on J10
- 6.00" x 7.5" board size

The system's most common configuration seems to be:

- System PCB
- 32M PC133 SDRAM DIMM
  - Unknown PCI IEEE-1394 card supported by OHCI driver
  - PPCBoot v.1.1.2 in flash

Hardware that is known to work (and not to work) with this board can be found here

GPSFan (Freenode IRC on #edev) was helpful in providing an image of the board:



Some info in the pdf HERE

Experimental hack to set a hardware breakpoint inside OCD Commander hardbreak.tgz

- OCD Commander macro file to dump the CPU configuration (all but PCI) dump_config.mac
- OCD Commander macro file to configure the system for recovery recovery_config.mac
- CPU configuration of the system after a warm reset warm_reset.config
- CPU configuration of the system once ppcboot v1.1.2 is up and running ppcboot.config
- CPU configuration of the system once ppcboot v1.1.6 is up and running ppcboot_116.config
- Example OCD logfile of the using the recovery macro to load ppcboot ocd_recovery_log.txt
- Example console log from ppcboot v.1.1.6 running from ram after recovery macro ocd_recovery_console_log.txt

Bootloaders

- ppcboot

Here is the latest ppcboot 1.1.6 patch, as is, it will build for flashing at 0xfff80000 as a replacement for the pcboot-1.1.2 that comes with the board.

patch-ppcboot-1.1.6-km2

Original sources for ppcboot: [1]

Here is a binary for those without a toolchain.

ppcboot1.1.6.1.bin crc = 083fb0a3

To use this (at your own risk) see the detailed update procedure description: Installing the Updated Bootloader

- U-Boot

U-Boot is a significantly updated replacement for ppcboot. See: U-Boot for the DHT-Walnut.

Just in case you make a brick, there is hope: JTAG Bootloader Installation

Linux Kernels

- Version 2.4

(Mostly) working kernel binary and config from jbevren with matrox fbcon and usb input support.

Current issues:

-
  - USB keyboard input doesnt seem to work, but events reach /dev/input/event0
  - Media:uImage
  - Media:config
- Version 2.6

See: 2.6 Linux kernel for the DHT-Walnut

Userland

- Debian GNU/Linux netboot installer -- Install Debian from the internet to a hard disk.

How-To's

- How To replace the bootloader with ppcboot 1.1.6
- How To replace the bootloader with U-boot 1.1.4
- How To recover from "Brickage"
- How To add the JTAG connector at J10
- How To boot via tftp
- How To boot from Hard Disk
- How To do something useful with your board
- How To control the Green Media LED and J5
- More to come...

Categories:

- Development Boards
- DHT-Walnut

From: eLinux.org

Didj

This device is part of the LeapFrog Pollux Platform, it's recommended to start there for general information.



The LeapFrog Didj

Contents

Summary

The Didj was a toy produced by Leapfrog marketed for educational games for children aged 5-10.

Didj was end-of-lifed by Leapfrog in mid-2010. It has been replaced by the Leapster Explorer.

Although Didj has a proprietary graphical front end, it runs a generic Linux distribution on an Arm based processor. Soon after the Didj's release, it was discovered that the cartridge port contained pins that allowed for serial console access with root privileges. After this discovery, work began to modify the Didj into an accessible emulation device.

Since development began, much has been accomplished, including:

- Discovered that the Arm chip is the same as on the GP2X Wiz, only at a lower clock speed.
- Created cartridges that support SD cards

- Accessed the UART features in the cartridge slot

Platform

LeapFrog Pollux Platform

The Didj is part of 3 different devices that all share a common hardware platform, based around the Pollux SoC. The platform page contains information generic across these devices, and it is recommended that you refer to that page as it is a good starting point to understanding the Didj, and contains some basic How To's and Tutorials to get you started.

Boot Loader

- Lightning Boot
- U-Boot
- Compile and Install Emerald-Boot

Sources and Toolchains

- Sources and Toolchains

Tutorials/How To's

General

- Common Commands Reference
- Console Access
- Cartridges
  - Cartridge Settings
- Extract lfp/lf2 Archives

Networking

- Enable Networking via USB Gadget
- Install Dropbear SSH
- Playing MP3 network streams and files
- Networking Setup
- Networking Applications
- Internet Access from Device
- Mount NFS Directory

USB Storage

- USB Mounting Under Windows, Linux, and OS X
- SCSI Commands

Cartridge

- Make an ATAP NAND Cartridge

Flash NAND

- Flash Data to NAND
- Updating Bootloader/Firmware

Firmware Image

- Mount JFFS2 Image on Linux
    - Relevant Settings
    - Correct endianess
- Create JFFS2 image
    - Relevant Settings
    - -e 128
    - -p

Kernel/RootFS/Firmware

- Building libSDL
- Building SDL_ttf font library
- Building SDL_image library with jpg and png support
- Building SDL_mixer audio library
- Building tslib
- SDL Resources
- Didj Kernel 2.6.31 Upgrade
- 2.6.31 Kernel for Didj
- Boot Kernel and Rootfs from SD w/Framebuffer
- Linux Framebuffer Driver
- Enable SD Card Module
- Building The Explorer Root File System
- Changing the fb driver to display the boot logo correctly
- TV Out
- Generic Buildroot Rootfs and Kernel build

Games and Emulators

- Emulators and Games

Brio Development

- Replacing the default App Menu from the default App Menu

JTAG

- JTAG Pinouts
- Pollux FTDI JTAG How To
- Pollux JTAG Wiggler Config
- JTAG Kernel Boot

Compiling Source Code

- Set up the Build Environment
- Kernel Configuration

Development Scripts and Programs

- LF1000 UART Bootstrap Utility written in Python
- LF1000 UART Bootstrap Utilities based on the OMAP boot utilities from TI

Technical Information

- Initial Memory Map Dump
- Map of Didj GPIO Pins

- Device Comparison

File System Info

- rootfs / rootfs rw
- /dev/root / jffs2 ro
- none /proc proc rw
- sysfs /sys sysfs rw
- /dev/ram0 /tmp tmpfs rw
- /dev/mtdblock1 /flags jffs2 rw,sync,noatime
- /dev/mtdblock2 /mfgdata jffs2 ro,sync,noatime
- /dev/mtdblock10 /Didj vfat rw,noatime,fmask=0022,dmask=0022,codepage=cp437,iocha rset=iso8859-1
- /dev/mtdblock11 /Cart vfat ro,noatime,fmask=0022,dmask=0022,codepage=cp437,iocha rset=iso8859-1

```
Filesystem                Size       Mounted on
/dev/mtdblock6           14.0M      /
/dev/mtdblock1          896.0k     /flags
/dev/mtdblock2            1.0M     /mfgdata
/dev/mtdblock9          215.8M     /Didj
```

Active Kernel/Rootfs

Near the begging of the boot message you should see one of two root options:

```
root=31:04
```

You are using mtd4: 00e00000 00020000 "Linux_RFS0" or

```
root=31:06
```

You are using mtd6: 00e00000 00020000 "Linux_RFS1"

Partitions

Name	Location	Size	Device	Notes
LF1000_uniboot	0x00000000	0x00020000	/dev/mtd0	Lightning Boot
Atomic_Boot_Flags	0x00020000	0x000E0000	/dev/mtd1	On NAND
Manufacturing_Data	0x00100000	0x00100000	/dev/mtd2	On NAND
Kernel0	0x00200000	0x00200000	/dev/mtd3	On NAND
Linux_RFS0	0x00400000	0x00E00000	/dev/mtd4	On NAND
Kernel1	0x01200000	0x00200000	/dev/mtd5	On NAND
Linux_RFS1	0x01400000	0x00E00000	/dev/mtd6	On NAND
Brio	0x02200000	0x0DE00000	/dev/mtd7	On NAND
EXT	0x10000000	0x10000000	/dev/mtd8	
Cartridge	0x00000000	0x10000000		On Cartridge NAND

Battery Compartments

The Didj has two identical battery compartments, the combined collection of batteries are wired in series.

- Battery Compartment Terminals:
  - Term 1 - Battery +
  - Term 2 - Temp Sensor + Wired to Pollux pins K20 / GPIOA 28 and K21 / GPIOA 29

- Term 3 - Temp Sensor -
- Term 4 - Battery -

Rechargeable Batteries

In addition to the terminals facing the Didj's contacts, there are another set of contacts facing away from the Didj. These contacts are duplicates used by the charging station. The batteries themselves are NIMH cells, producing around 2.5v per pack when fully charged.

Recharging Station

The recharger station contains a battery charge circuit with temperature monitoring. The station also has a pair of contacts that duplicate the 9V dc barrel jack.

SSP / SPI Controller Didj SPI Info

Images

- PCB Images
- Error creating thumbnail: Invalid thumbnail parameters

  Front side mainboard, with LF1000 CPU and SDRAM de-soldered.

- 

  Close up of the LF1000 CPU ball-grid.

- Error creating thumbnail: Invalid thumbnail parameters

  Back side mainboard, with cartridge socket and NAND de-soldered.

- 

  Front side mainboard

- 

  Back side mainboard

External Links

- Cozybit boasts of their involvment in integrating Linux with the Didj
- Hacking around with the Leapfrog Didj
- Claude's Didj Hacking Page
- Hackaday Posts

    - Didj Hacking
    - Didj Hacking Followup

Categories:

- Didj
- LeapFrog Pollux Platform

> From: eLinux.org

EBR-1000EP

Sony Librié

http://www.eink.com/news/images/SONY_Reader_1000EP.jpg

This is Tim Riker's space for taking notes. Much of this is copied from elsewhere on the net including the Yahoo Librié group.

- Motorola MC9328MX1 (ARM920T) http://www.freescale.com/files/32bit/doc/data_sheet/MC9328MX1.pdf
- 6in reflective electronic paper display with E Ink technology (2bit grayscale)
- SVGA (800x600 dots) resolution at 170dpi
- 300g with case and batteries (190g without)
- 126mm x 190mm x 13mm
- Sony Linux OS: NSC Linux version 1.2.0 Patch 4.1
- 64MB SDRAM system memory (62500KB free)
- 4MB NOR flash memory (containing Linux bootloader, kernel, Memory Stick BIOS, keys and device informations)
- 32MB NAND flash memory Samsung KM29U256T? (containing fonts, dictionaries and bookshelf data, 11152KB free)
- 16MB NAND flash memory Samsung KM29U128T? (containing rootfs and Open MG data)
- Memory Stick slot
- USB 2.0 port
- Headphone jack
- Rear-mounted mono speaker
- 4xAAA batteries, ~10.000 pageturns
- Miniature Qwerty keyboard

Firmware

- http://www.aii.co.jp/contents/smojsdmk/LIBRIE/UPLIBRIE_06160.EXE (2004.06.23)
- http://www.aii.co.jp/contents/smojsdmk/LIBRIE/UPLIBRIE_04140.EXE (2005.04.13)

Links

- Runs Linux: http://www.sony.net/Products/Linux/Download/EBR-1000EP.html
- Appears to be a MV build
- Tear Down: http://www.teardown.com/channels/pdas/Sony_Librie.aspx
- 100 pin test connector CN1501 likely a Molex 54137-1008 http://www.molex.com/customer.html?supplierPN=541371008
- mate should be 53929-1008 http://www.molex.com/customer.html?supplierPN=539291008
- http://www.karpenko.de/librie/
- http://www.positron.org/projects/juicebox/
- http://groups.yahoo.com/group/librie/
- http://groups.yahoo.com/group/cybook/
- http://www.flickr.com/photos/deadsunrise/sets/444695/
- http://www.dynamism.com/librie/
- http://www.excite.co.jp/world/english/web/
- http://www.sven.de/librie/
- news://news.gmane.org/gmane.linux.hardware.sony.librie
- http://news.gmane.org/gmane.linux.hardware.sony.librie

- http://www.timebooktown.jp/
- http://www.get-set.net/librie/ - translated buttons
- http://en.wikipedia.org/wiki/E-book
- http://www.dzhabrailov.ru/71417.html
- http://greggman.com/japan/xp-ime/xp-ime.htm
- http://www.rikai.com/library/kanjitables/kanji_codes.euc.shtml
- http://isthisthingon.org/unicode/
- http://www.sony.jp/products/Consumer/LIBRIE/download/libriele.html
- http://www.mynetcologne.de/~nc-bergme2/librie.htm
- http://rts-lab.eas.asu.edu/courses/cse494f03/images/userguide.pdf
- http://developer.berlios.de/projects/librietrans/

Category:

- Hardware Hacking

> From: eLinux.org

Enc28j60

Introduction

Microchip's `ENC28J60` is a 28-pin, 10BASE-T stand alone Ethernet Controller with on board MAC & PHY, 8 Kbytes of Buffer RAM and an SPI serial interface. With a small foot print package size the `ENC28J60` minimizes complexity, board space and cost. Target applications include VoIP, Industrial Automation, Building Automation, Home Control, Security and Instrumentation.

Evaluation Boards

- Olimex Board
- edtp.com

The olimex board is also sold at Sparkfun

Linux Drivers

- 2.4 Kernel Driver , this an experimental driver to test the basic features of the enc28j60 under linux. it was done using the s3c2410 arm920t. this driver can be modified to work with other spi controllers including bitbanging. this is a VERY ugly hack and is not ment to be a production driver.
- 2.6 Kernel Driver no SPI subsystem , this is the initial driver support for the enc28j60. this driver uses a spi interface header that describes the device being used. (Completed, waiting on lawyers approval for release)
- 2.6 Kernel Driver with SPI Subsystem, this version of the driver supports the new SPI subsystem that is available from 2.6.16 and newer. (ETA 8/28/2006 update: there is a driver in the Linux Kernel
  - .../drivers/net/ethernet/microchip/enc28j60.c)

References

- Microchip ENC28J60 Page
- http://elinux.org/images//8/83/Pdf.gif Microchip ENC28J60 Datasheet http://elinux.org/images/d/da/Info_circle.png
- Microchip ENC28J60 Errata
- AVRlib with ENC28J60 support

if you are using the enc28j60 and would like to contribute changes/fixes/updates please send them to prpplague

Category:

- Hardware Hacking

> From: eLinux.org

Ez Usb

Contents

EZ-USB Based Device List

Firmware Kit + Driver API

An Open Source EZ-USB SDK is available at http://www.ztex.de/firmware-kit.

It is mainly intended for our ZTEX FPGA Boards but should run on every EZ-USB based hardware. The package currently runs under Linux and Windows.

The Firmware is written in C and requires the SDCC compiler. A macro approach allows to configure the Firmware easily using macro commands. The necessary USB descriptors and the descriptor handling routines are generated automatically.

The driver API is written in Java and allows platform independent device drivers. Alternatively the USB devices can be accessed through a web and a socket interface using the device server.

General Products

- Orinoco Wireless USB Adapter
- HH501 Smartmedia Reader/Writer
- Keyspan USB-to-DB9-serial Adaptor
- Orange Micro iBOT2 USB 2.0 Web Camera
- Flash2Advance Ultra USB Writer

Development Boards

- DeVaSys USB I2C/IO EZ-USB board
- ActiveWire USB Devices
- USB Interface V1.3 with EEPROM
- MF3001 EZ-USB AN2131QC Development Board
- OSR USB FX2 Learning Kit
- USBSIMM SimmStick Development
- EZ Protoboard
- Quick USB Module
- High-Speed USB Module
- ZTEX USB-FPGA-Module 2.16: FPGA Board with EZ-USB Microcontroller and Artix XC7A200T FPGA
- ZTEX USB-FPGA-Module 2.13: FPGA Board with EZ-USB Microcontroller, Artix XC7A35T to XC7A100T FPGA and 256 MB DDR3 SDRAM
- ZTEX USB-FPGA-Module 2.04: FPGA Board with EZ-USB Microcontroller, Spartan 6 XC6SLX16 FPGA and 64 MB DDR SDRAM
- ZTEX USB-FPGA-Module 2.01: FPGA Board with EZ-USB Microcontroller and Spartan 6 XC6SLX16 FPGA

Category:

- Development Tools

> From: eLinux.org

Flameman

- Flameman/project
- Flameman/article (it needs re-arrangement, proof notes)

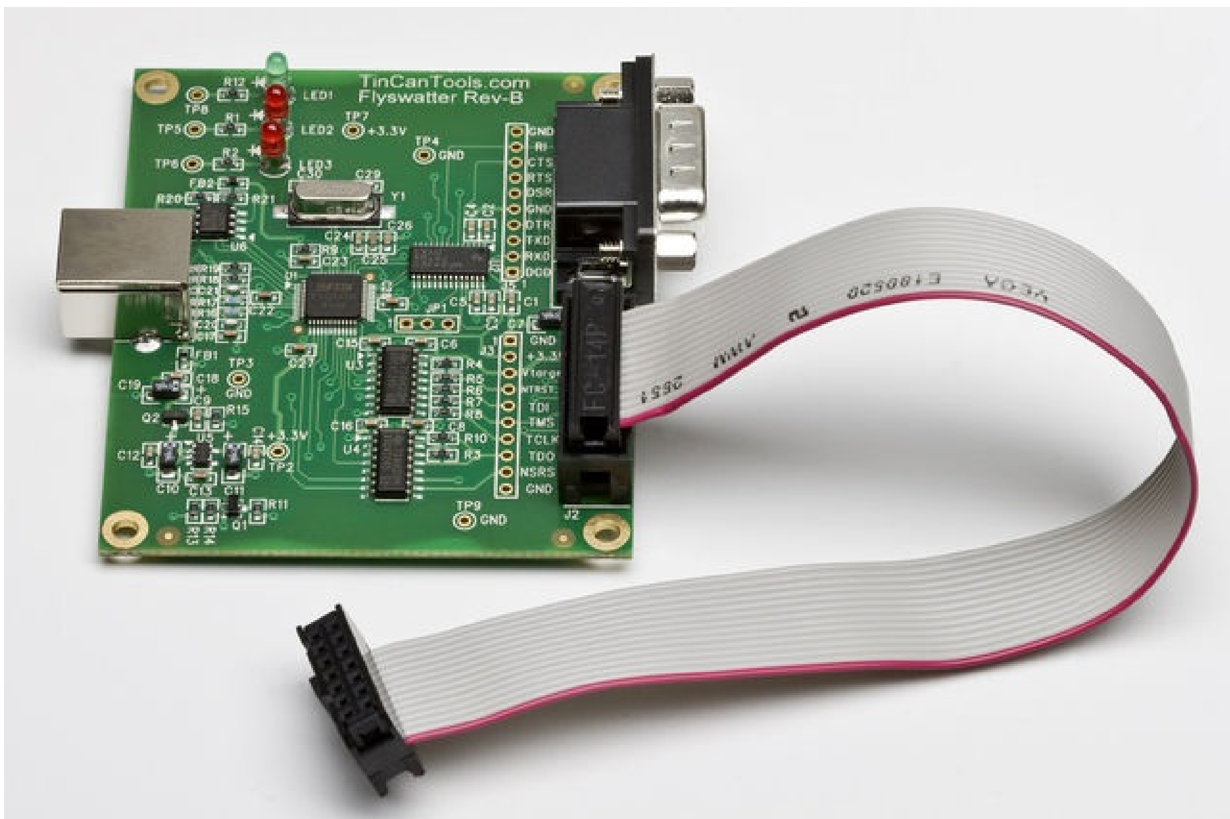
From: eLinux.org

Flyswatter

Contents

Introduction

The Flyswatter JTAG board is a low cost USB based JTAG programmer for the Hammer CPU board. It can be used with all ARM processors that are supported by OpenOCD. It is based upon FTDI's popular FT2232 USB UART/FIFO IC. The Flyswatter provides a standard JTAG interface as well as a standard RS232 port with support for full modem signals.



available from TinCanTools at http://www.tincantools.com/JTAG/Flyswatter.html

Features:

- USB 2.0 Full Speed device (12 Mbits/sec) interface
- Can be used to debug and program all ARM processors supported by OpenOCD
- Provides a standard 2x7 pin JTAG interface
- Adds a virtual RS232 serial port to your computer or laptop with full modem signals: TXD, RXD, RTS, CTS, DTR, DSR, DCD, RI

- Supports ARM target voltages of: 3.3V, 2.5V, 1.8V, 1.5V, 1.2V (voltage range: 1.2V to 3.6V)
- Adds a virtual RS232 serial port to your computer or laptop with all of the modem signals: DTR, DSR, DCD, RTS, CTS, Rx, Tx
- No external power supply required – it runs off of the USB voltage from the computer
- Open hardware – complete schematic provided
- Open software - software supported by OpenOCD (open source) debugger
- Included CD comes with OpenOCD for Linux
- Dimensions: 2.5 inches (width) x 3.0 inches (height)
- Package Includes: Flyswatter board, USB Cable, 8 inch JTAG ribbon cable (14 pin - 2x7)

Supported Devices

OpenOCD supports the following ARM cores:

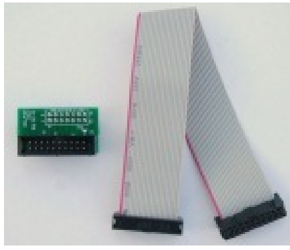
ARM CORE	EXAMPLE PROCESSORS
ARM7TDMI	LPC2148, AT91SAM7
ARM720T	LH79520, EP7312
ARM9TDMI	
ARM920T	S3C2410, S3C2440
ARM922T	
ARM926EJS	S3C2412, STN8811, STN8815
ARM966E	STR91XF
ARM11	S3C6400, OMAP2420, MSM7200
ARM1136	
ARM1156	
ARM1176	
CORTEX-M3	LM3S series, STM32 series
CORTEX-A8	OMAP3530 BeagleBoard
CORTEX-A8	DM3730 BeagleBoard-xM
CORTEX-A9	OMAP4430 PandaBoard
XSCALE	PXA255, PXA270, IXP42X
MARVEL	FEROCEON CPU CORE

OpenOCD also supports the following MIPS cores (requires a MIPS 14-Pin JTAG Adapter):

MIPS CORE	EXAMPLE PROCESSORS
MIPS	M4K

JTAG Adapters

JTAG Adapters plug into the ARM 14-pin connector located on the Flyswatter and change the pin-out to a different JTAG interface. Three JTAG adapters are available for the Flyswatter:



ARM 20 pin JTAG Adapter

ARM 20-Pin JTAG Adapter -This adapter converts the Flyswatter's JTAG interface into a standard ARM 20-pin configuration. The package also includes 14-pin ribbon cable.



BeagleBoard JTAG Adapter

BeagleBoard JTAG Adapter -This adapter converts the Flyswatter's JTAG interface into a standard TI 14-pin JTAG configuration. The package also includes a serial adapter board that converts the DB-9 Male connector located on the Flyswatter to a 10-Pin ribbon cable. This ribbon cable is used on the BeagleBoard Rev B/C boards to interface to the serial port. The 10-pin ribbon cable is included in the package.



MIPS 14-Pin JTAG Adapter

MIPS 14-Pin JTAG Adapter -This adapter converts the Flyswatter's JTAG interface into a standard MIPS 14-pin JTAG configuration. The package also includes 14-pin ribbon cable.

Serial Port

The Flyswatter's serial port provides you with an independent functional "USB to RS-232" serial device. The serial port is completely independent from OpenOCD on both Linux and Windows. You can use the Flyswatter's serial port and never have to use OpenOCD or JTAG, or you can use it together with OpenOCD and have both a serial port and JTAG interface operating at the same time for debugging your target device.

For Linux, the RS232 driver for the FT2232 is part of the main kernel tree and is provided in most standard Linux distributions. In Windows, you have to load the Windows driver for the FT2232. Once the driver is loaded, Windows will assign a virtual COM port to the Flyswatter's serial port. It operates just like a standard COM port. You can use the Flyswatter's serial port on laptops or PC's that do not have a 9-pin legacy serial connector.

Serial Port Interface

You can use Minicom to communicate with the Flyswatter's serial port on Linux. See the Minicom page for setup instructions.

Documents:

- JTAG Connector Pin Assignments
- Flyswatter schematic diagram

Source Code:

- OpenOCD Source
- LibFTD2xx library files

available from TinCanTools

Devices that have been tested:

- Hammer: Supported by the software kit/tools that come with the Hammer
- NSLU2: requires a JTAG connection to be added to the NSLU2 and nslu2.cfg, nslu2.ocd and Debug_handler.bin files. You can use these samples, but may need to edit the nslu2.ocd as appropriate.
- Zipit Z2: requires soldering directly to the Z2 board, and using z2.cfg and following this how-to.

Flyswatter How To

The Flyswatter How To and Flyswatter Windows How To guides show a new user how to connect the Flyswatter to the Beagleboard, and how to install and run OpenOCD and GDB Debugger. To reach the guide, follow the link in the section title.

Flyswatter2

TinCanTools has a new JTAG debugger at Flyswatter2 that has support for OpenOCD and ARM Cortex A8 processors: OMAP3530 BeagleBoard and DM3730 BeagleBoard-xM. The Flyswatter2 is 5 to 10 times faster than the original Flyswatter.

They also have a ARM20TI14 JTAG Adapter. This JTAG adapter board works with the BeagleBoard and BeagleBoard-xM.

The Flyswatter2 can be used with OpenOCD (Beagle (OMAP3xx) support is complete).

To use the Flyswatter2 under Ubuntu (and derivatives) without superuser rights, place this in /etc/udev/rules.d/60-flyswatter2.rules (or whatever name you like):

```
SUBSYSTEM=="usb", ACTION=="add", ENV{DEVTYPE}=="usb_device", \
SYSFS{idVendor}=="0403", SYSFS{idProduct}=="6010", MODE="0666"
```

Maybe it also works for the original Flyswatter, but not yet verified. If not, the idVendor and idProduct values must be adjusted.

Categories:

- TinCanTools
- Flyswatter

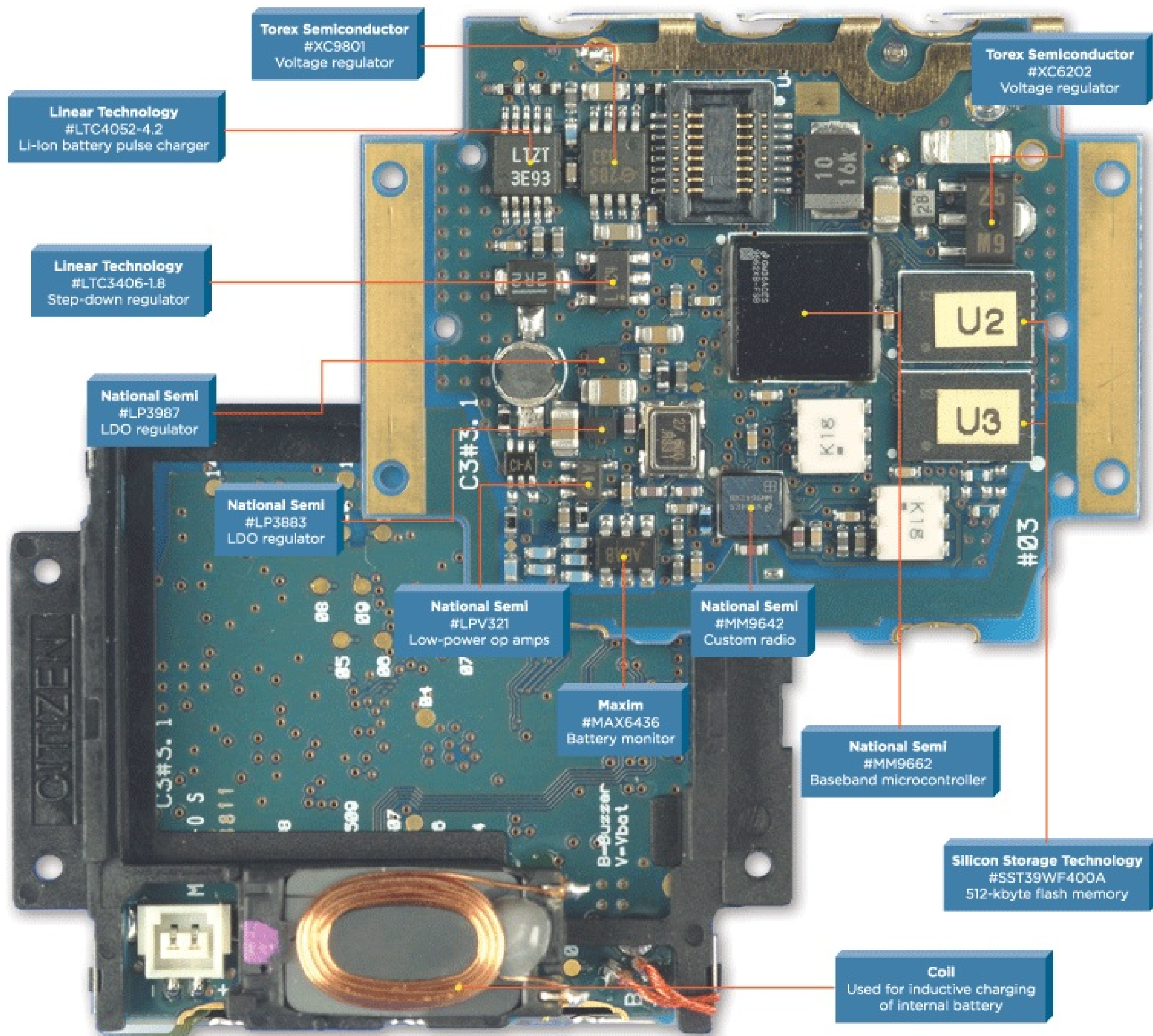
> From: eLinux.org

FX3002



A development effort between Microsoft and National Semiconductor Corp. gave rise to the key ICs for the Spot engine. The processor chip (National's MM9662) contains an ARM7 CPU and needed ROM/SRAM and is supported by two 512-kbyte flash memory packages from Silicon Storage (SST39WF400A). National also supplies the custom radio chip (MM9642) receiver along with op amps (LPV321) and low-dropout regulators (LP3987, LP3883). A lithium-ion battery pulse charger (LTC4052-4.2) made by Linear Technology; a Maxim battery monitor (MAX6436); and two Torex voltage regulators (XC6202, XC9801) round out the power-management circuitry. Citizen Watch of Japan was chosen for the module manufacture.

With the rich set of electronics and always-on nature of the device, low-power design was a must given the internal 3.7-volt, 150-mA-hr Li-ion battery. Extending the wireless theme, a clever induction charger is used to ac-couple fresh juice through the hanging watch stand.



Category:

- Hardware Hacking

> From: eLinux.org

Hello World in C

```
#include <stdio.h>

main() {

for(;;)
    {
    printf ("Hello World!\n");
    }

}
```

Categories:

- Development Tools
- C

> From: eLinux.org

Hisense



US Digital (USDTV http://usdtv.com/) sold an HDTV Receiver called the DB-2010 manufactured by HiSense. The unit runs an embedded Linux 2.4.18-12 kernel, with usb-storage compiled into the kernel. They swap MTD flash RAM partitions 1 and 4 with each new firmware update (similar to Tivo).

- http://www.walmart.com/catalog/product.do?product_id=2598451
- http://www.hisense.com/english/ - manufacturer
- http://usdtv.com/ - reseller
- http://www.ati.com/companyinfo/press/2003/4628.html
  - related press release?
- http://www.linuxdevices.com/products/PD4781682328.html
  - Xilleon 220?
- http://rikers.org/gallery/hardware-hisense
- http://linux-hacker.net/usdtv - serial port pinout dmesg and boot
- http://web.archive.org/web/20061019225229/http://www.videohelp.com/forum/archive/t302016.html
- http://www.avsforum.com/avs-vb/showthread.php?t=479882
  - active forum
- http://www.vanillahd.com/index.php/receiving-free-over-the-air-ota-tv-with-a-usdtv-box/
  - more unlocking info
- http://www.avsforum.com/avs-vb/showthread.php?t=1055192
  - hardware versions and other useful bits
- http://www.mediafire.com/dogleg69
  - patched firmware including DST fix and timers

The box has a card slot on the front, but there is no card reader behind the slot. It has a single USB port on the back that can be used to upgrade the firmware. This uses the AMD/ATI Xilleon 225, a MIPS processor.

Tim Riker wrote USDTV requesting the source code on 2006.03.22 and got this reply on 2006.03.31:

```
Dear Mr. Riker,

Your request for access to the GPL and LGPL source code used in the USDTV DB-2010
receiver was recently forwarded to me.  First of all, let me assure you that we
at USDTV are fully aware of the terms of these software licenses.  Many members
of our development team are long-time users and supporters of free and
open-source software.

As you have requested, we will make available for Internet file transfer copies
of the software used in the USDTV receiver that is covered by these licenses.
Unfortunately, your request has caught us at a bad time.  The USDTV development
offices are currently in the process of moving to a new location, so we do not
at this time have a server to host copies of the software to download.  Once our
move is completed and our full Internet service is restored, we will set up a
site with the requested software available.  In the meantime, I have attached
below a list of software that we use; all of these packages should be available
for download elsewhere on the Internet.

   Linux kernel version 2.4.18
   glibc version 2.2.4
   libpthread version 0.9
   busybox version 0.60.0
   GNU tar 1.13.19
   gzip version 1.3

Thanks,
Jim Burmeister
Technical Lead, Set-Top Box Software Development
U.S. Digital Television
```

Some other emails have gone back and forth, but no sources have yet been made available to Tim Riker.

USDTV filed for Chapter 7 bankruptcy in the US District Court of Delaware, and approximately 6600 receivers (many new in box) were sold on **July 17th 2007 at 10:00AM** at 12552 S 125 West, STE 200 Draper, UT (upstairs @ DAW Group Bldg).

- http://www.salesandauction.com/7172007.htm
- http://www.salesandauction.com/usdlotlist.htm

For those of you that never received a firmware update, here is the update. It contains GPL code yet we have no method of extracting only that code. As the GPL portions are freely copyable, and not extractable, the package itself must also be copyable. Were it not so, the GPL would be violated, and I'm sure they would never do that. With no further ado, here are the packages. **REMEMBER TO NOT INCLUDE THE DELETEME.txt ON THE USB STICK** or the upgrade **WILL FAIL**.

- File:USDTV-2.7.11.tgz
  - last known update for non-subscription (ie: FTA or WalMart) boxes
- File:USDTV-2.7.15-FTA.tgz
  - last known update for subscription (ie: USDTV) boxes to use over the air

The two different installed firmware images appear to use different digital signatures. This is why one firmware update will not run on the other machine. It is expected that they can be swapped, but we are not publishing this information yet. We expect to publish extracted filesystem images here soon.

File signatures compatible with the 2.7.15-FTA upgrade are computed by obtaining the 32-character MD5 checksum of the file, then performing an MD5 hash using the salt *1@M&k=Ba}$* or '$1$69-1@4Pm$'

The low-level boot code is ATI's (now AMD's) proprietary version of MMON, very likely derived from Eric Smith's mmon MIPS monitor - http://www.brouhaha.com/~eric/software/mmon/

If you have other GPL-containing firmware updates you would like posted here, please contact Tim Riker.

Hacks

- Guide-Aspect-Exit-Aspect-Guide - turns on/off a setup -> diagnostics menu including "stream info"
- Guide-Aspect-Exit-ProgramInfo-0 - factory reset (do we need the 0 ?)
- Guide-ProgramInfo-Exit-Aspect-Exit - Update Configuration - change when box checks for updates that will never come
- Guide-ProgramInfo-Aspect-ProgramInfo-1 - reboot to other rom image (delays 17 seconds or so before reboot)

Category:

- Hardware Hacking

> From: eLinux.org

Industrial Communications

Plants and machinery today consists of networks of different kind of components each are communicating with each other. The communication protocols used are partly Ethernet based, or traditional two wire buses (field bus) like CAN.

- CAN Bus
- Ethernet POWERLINK
- Ethernet/IP
- PROFINET

Category:

- Networking

> From: eLinux.org

InnoTab



Contents

Description

Take learning to the next level with the InnoTab Learning App Tablet by VTech! This multi-media tablet combines interactive reading, learning games, creative activities, and a rich collection of applications into a sleek and durable educational toy that kids will want to play with. Featuring a 5" color touch screen and tilt-sensor, kids can tap, flick, drag, and pat their way to learning fun. The fun continues with many on-board applications such as an MP3 Player, Video Player, Art Studio, Friends List, Calculator, and Clock engaging kids for hours as they develop their skills with this electronic learning toy. Additional cartridges with favorite licensed characters are sold separately and teach essential skills in reading, logic, and creativity. Additional content such as e-books and learning games can easily be uploaded to the InnoTab through VTech's Learning Lodge Navigator where parents can also see their child's progress on a variety of educational milestones and lessons. Additional content may require an SD memory card (not included).

Specifications

- GPL32902 processor
  - ARM11 @ 180MHz
  - Manufacturer GeneralPlus
  - GPL3900 Development Platform
- 64MB onboard memory
- SD card slot for memory expansion (supports SD/SDHC memory cards between the sizes of 2GB and 16GB.)
- 5" Color LCD
- Touchscreen
- Tilt sensor
- uses 4AA batteries

CPU info

```
Processor       : ARMv6-compatible processor rev 7 (v6l)
BogoMIPS        : 359.62
Features        : swp half thumb fastmult vfp edsp java
CPU implementer : 0x41
CPU architecture: 7
CPU variant     : 0x0
CPU part        : 0xb76
CPU revision    : 7
Hardware        : GPL32900
Revision        : 0000
Serial          : 0000000000000000
```

Source Code

- git repo of the build system release
- tar ball of the build system release

Hacks

- Boot Log
- add an external 3.5mm serial port



- information from user "BT-Vision" aka "is0-mick" on wizardmods.net forums

```
The innotab uses a Sqlite database to keep track of its installed applications (this is copied to the SD card when a
blank one is inserted).

Icons are also stored in the SQLite database as a binary blob.
Game icons are 57 x 57, book icons are 88 x 88

They are 16bit xRGB data, with the first 8 bytes being the width and height stored as 32bit integers in little endian
 format.

If you just want to run your own code you can do the following.
Download SQLite manager extension for firefox, use it to open innotab.db which is on the SD card.

Go to the “games” section, and create a duplicate entry of one that already exists..

Change the filename on the new entry to something like /vp_mnt/sd/LLN/APPS/myapp

create a folder called on the as card /LLN/APPS/myapp

in that folder create a file called _Run
and put shell commands in there :) and the innotab will execute them.

for instance try ls -aR / >/vp_mnt/sd/LLN/APPS/myapp/dir.txt
```

- Initial porting scummvm to the Innotab can be seen here:

Resources

- wizardmods.net forum on InnoTab hacking
- Hackaday post on InnoTab
- VTech InnoTab Product Page

> From: eLinux.org

JuiceBox



uClinux media player from Mattel made by Emsoft

- Hardware Specifications
- Software Specifications
- General JuiceBox Info

NEWS FLASH it is now possible to boot a custom uclinux kernel and ramdisk

- SD/MMC Cartridge Hack

NOTE:

for those who are viewing this page for the first time and are considering hacking the **JuiceBox** you should be aware of some facts. my primary reason for hacking the **JuiceBox** was not to make a mp3 player, picture viewier, or movie player. i never ment for the **JuiceBox** hack to be anything other than an inexpensive platform to experiment with design concepts. this includes experimentation with NAND flash, NOR flash, arm7tdmi architecture, bootloaders, SD/MMC card interfaces, and uClinux. if your intentions for hacking the **JuiceBox** are for anything other than education, you will probably be disappointed. -prpplague

Hacks

- http://www.flickr.com/photos/nicrosin/sets/1162716/
- http://www.flickr.com/photos/tv/6694555/in/pool-make/
- http://www.flickr.com/photos/75872235@N00/sets/72057594056009141/
- http://www.hackaday.com/entry/1234000203045338/
- http://myvogonpoetry.com/wp/index.php?p=57
- http://www.bigmech.com/misc/juiceboxhack/juiceboxhack.html

Lesser Hacks

The discontinuation of sales of the **JuiceBox** moves some previously trivial applications into the realm of *lesser hacks*; things that require some degree of documentation and effort, but no actual hardware modifications of the **JuiceBox** itself.

- JuiceBox Music Player Play MP3 songs on your **JuiceBox**
- JuiceBox Picture Player Show Pictures on the **JuiceBox** screen
- JuiceBox FAQ Hard Things things you might have hoped were easy, but aren't
- JuiceWare_Xd_Proc custom cartridge

Category:

- JuiceBox

> From: eLinux.org

LeapPad Explorer

(Redirected from Leappad Explorer) Redirect page

Redirect to:

LeapPad1 Explorer

> From: eLinux.org

Leapster

Technical Specifications

- CPU: Custom ASIC containing an ARCTangent 5.1 CPU, running at 96MHz.
- Memory:
  - Original Leapster: 2MB onboard RAM, 256 bytes non-volatile.
  - Leapster2: 16MB RAM, 128kbytes non-volatile storage
- Media type: Cartridges of 4-16MB with between 2 and 512kb non-volatile storage.
- Graphics: No hardware acceleration.
- Audio: Proprietary hardware audio acceleration.
- Screen: 160x160 CSTN with touch sensing.

All of the software content for the original Leapster was created with Macromedia Flash MX 2004; the device runs an embedded Macromedia Flash player.

Category:

- Leapster Explorer

> From: eLinux.org

Leapster Explorer

This device is part of the LeapFrog Pollux Platform, it's recommended to start there for general information.



The Leapster Explorer (Leapfrog Enterprises)

Contents

Platform

LeapFrog Pollux Platform

The Leapster Explorer is part of 3 different devices that all share a common hardware platform, based around the Pollux SoC. The platform page contains information generic across these devices, and it is recommended that you refer to that page as it is a good starting point to understanding the Leapster Explorer, and contains some basic How To's and Tutorials to get you started.

Project Summary

The Leapster Explorer (like its predecessor the Didj) is a toy produced by Leapfrog marketed as an educational handheld gaming console for kids aged 4-9. Although it has a proprietary graphical front end, it runs a generic Linux distribution on the same ARM9-based processor as the Didj. Console access to the Explorer is achieved using the same method as found on the Didj.

Lots more information on the Didj

Bootloader

- Emerald Boot

Sources and Toolchains

- Sources and Toolchains

Tutorials/How To's

General

- Common Command Reference
- Overclocking
- Console Access
- Cartridges
    - Cartridge Settings
- Extract lfp/lf2 Archives
- CBF File Format
- DFTP

Booting

- GPIO Subsystem Boot Options
- USB Booting
- USB Boot without LFConnect
- Modify Kernel for USB Boot
- USB Boot Settings
- Surgeon

Networking

- Networking Setup
- Networking Applications
- Internet Access from Device
- Mount NFS Directory

LeapFrog Flash UI

- Play Your Own Theora Video Files
- Play Your Own Flash Game Files

Cartridge

- How to make a NAND cartridge

Kernel/RootFS/Firmware

- Building libSDL
- Building SDL_ttf font library
- Building SDL_image library with jpg and png support
- Building SDL_mixer audio library
- Building tslib
- SDL Resources
- Boot Kernel and Rootfs from SD w/Framebuffer
- Linux Framebuffer Driver
- Enable SD Card Module
- Building The Explorer Root File System
- Changing the fb driver to display the boot logo correctly

- TV Out

Firmware Image

- Mount UBI Image on Linux
  - Relevent Settings
  - ID Bytes 0x2C 0xDC 0x00 0x15
- Create UBI Image on Linux
  - Relevant Settings
  - vol_name=ubi_rfs
  - vol_size=87349248
  - -m 2048
  - -e 129024
  - -c 677
  - -s 512
  - -O 512

JTAG

- JTAG Pinouts
- Pollux FTDI JTAG How To
- Pollux JTAG Wiggler Config
- JTAG Kernel Boot

Compiling Source Code

- Set up the Build Environment
- Kernel Configuration

Technical Information

Device Comparison

Leapster Explorer Memory Map

Camera Interface

File System Contents List

1.0.8.6905

1.1.46.8291

1.3.4.2044

1.4.11.2128

Default Boot CommandLine

```
init=/sbin/init console=ttyS0,115200 mem=46M mlc_fb=0x82E00000,0x01200000 ram=0x80000000-0x83FFFFFF screen_module=ILI
9322
ebs=0x20000,0x0 root=ubi0_0 rw rootfstype=ubifs ubi.mtd=RFS
```

Partitions

Name	Location	Size	Device	Notes
Emerald_Boot	0x00000000	0x00100000	/dev/mtd0	On NOR
I18n_Screens	0x00100000	0x00100000	/dev/mtd1	On NAND
Kernel	0x00200000	0x00800000	/dev/mtd2	On NAND
RFS	0x00A00000	0x05600000	/dev/mtd3	On NAND
Bulk	0x06000000	0x1A000000	/dev/mtd4	On NAND
NOR_Boot		0x0007E000	/dev/mtd5	
MfgData0		0x00001000	/dev/mtd6	
MfgData1		0x00001000	/dev/mtd7	
ubi_rfs		0x0534d800	/dev/mtd8	
ubi_bulk		0x194f2800	/dev/mtd9	
Cartridge	0x00000000	0x10000000		On Cartridge NAND

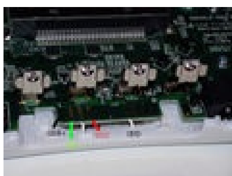
Images

- PCB Images
- 

  Scan of front side of Leapster Explorer mainboard.
- Error creating thumbnail: Invalid thumbnail parameters

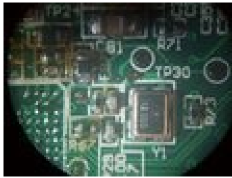
  With components desoldered - scan of front side of Leapster Explorer mainboard.
- 

  Scan of back side of Leapster Explorer mainboard.
- Error creating thumbnail: Invalid thumbnail parameters

  With components desoldered - scan of back side of Leapster Explorer mainboard.
- 

  Leapster Explorer USB Host Pinout. http://wtfmoogle.com/?p=1190
- 

  Leapster Explorer USB Host Connector.

- 

  TVout BGA pin(2A) and suspected test pad (TP30 right).

Categories:

- Leapster Explorer
- LeapFrog Pollux Platform

> From: eLinux.org

Libertas SDIO

Contents

Libertas SDIO

WPA

wpa_supplicant-0.5.5 is known to work properly with libertas sdio

Firmware 8.73.7

the marvell supplied firmware for 8.73.7 is known to work with all current versions of the libertas driver

http://www.marvell.com/drivers/driverDisplay.do?driverId=178

Firmware 9.70.3

the marvell supplied firmware for 9.70.3 is known to work with libertas driver in the main linux kernel starting at 2.6.27

http://www.marvell.com/drivers/driverDisplay.do?driverId=203 (dead)

http://extranet.marvell.com/drivers/driverDisplay.do?driverId=203

> From: eLinux.org

Literati

Contents

Links

1. New Dev forum http://literati.linuxprogrammer.org
2. Old Dev forum http://forum.literatidevs.com/index.php

Hardware

- CPU: S3C6410 running at 666Mhz
- Memory: 64MB
- LCD: innolux lcd 480 X 800

NAND (internal flash)

```
NAND device: Manufacturer ID: 0xec, Chip ID: 0xdc (Samsung NAND 512MiB 3,3V 8-bit)
Creating 5 MTD partitions on "NAND 512MiB 3,3V 8-bit":
0x00000000-0x00040000 : "Bootloader"
0x00040000-0x00340000 : "Kernel"
0x00340000-0x00d40000 : "resource"
0x00d40000-0x11140000 : "File System 1"
0x11140000-0x20000000 : "User space"
```

Datasheets

smdk6410 it looks to be based on this. File:Lookbook-schematics.pdf

LCD.pdf) quick view

JTAG Port

The JTAG port has yet to be identified on the Literati.

Serial Port

Serial Port Access

Opening Your Literati (White Sunway Model)

Opening White Literati

Kernel testing

Burn Kernel/Booting from SD

dmesg from ygator

```
Linux version 2.6.24.2 (root@ligang-laptop) (gcc version 4.1.2) #180 Mon Aug 23 00:49:07 CST 2010
CPU: ARMv6-compatible processor [410fb766] revision 6 (ARMv7), cr=00c5387f
Machine: SMDK6410
Ignoring unrecognised tag 0x00000000
Memory policy: ECC disabled, Data cache writeback
On node 0 totalpages: 16384
DMA zone: 128 pages used for memmap
DMA zone: 0 pages reserved
DMA zone: 16256 pages, LIFO batch:3
Normal zone: 0 pages used for memmap
Movable zone: 0 pages used for memmap
CPU S3C6410 (id 0x36410101)
S3C6410: core 666.000 MHz, memory 111.000 MHz, peripheral 55.500 MHz
S3C6410: EPLL 192.000 MHz
S3C64XX Clocks, (c) 2007 Samssung Electronics
CPU0: D VIPT write-back cache
CPU0: I cache: 16384 bytes, associativity 4, 32 byte lines, 128 sets
CPU0: D cache: 16384 bytes, associativity 4, 32 byte lines, 128 sets
Built 1 zonelists in Zone order, mobility grouping on. Total pages: 16256
Kernel command line: console=ttySAC0,115200 loglevel=0 ubi.mtd=3 root=ubi0:rootfs rootfstype=ubifs
Trying to install chained interrupt handler for IRQ0
Trying to install chained interrupt handler for IRQ1
Trying to install chained interrupt handler for IRQ32
Trying to install chained interrupt handler for IRQ33
PID hash table entries: 256 (order: 8, 1024 bytes)
timer tcon=00600900, tcnt 29b0, tcfg 00001919,00000040, usec 000077ee
Console: colour dummy device 80x30
console [ttySAC0] enabled
Dentry cache hash table entries: 8192 (order: 3, 32768 bytes)
Inode-cache hash table entries: 4096 (order: 2, 16384 bytes)
Memory: 64MB = 64MB total
Memory: 61708KB available (2768K code, 265K data, 104K init)
Calibrating delay loop... 665.19 BogoMIPS (lpj=1662976)
Mount-cache hash table entries: 512
CPU: Testing write buffer coherency: ok
net_namespace: 64 bytes
NET: Registered protocol family 16
================> wifi status change from 0 to 0
S3C6410 Power Management, (c) 2008 Samsung Electronics
s3c6410: Initialising architecture
S3C DMA-pl080 Controller Driver, (c) 2006-2007 Samsung Electronics
Total 32 DMA channels will be initialized.
SCSI subsystem initialized
usbcore: registered new interface driver usbfs
usbcore: registered new interface driver hub
usbcore: registered new device driver usb
NET: Registered protocol family 2
IP route cache hash table entries: 1024 (order: 0, 4096 bytes)
TCP established hash table entries: 2048 (order: 2, 16384 bytes)
TCP bind hash table entries: 2048 (order: 1, 8192 bytes)
TCP: Hash tables configured (established 2048 bind 2048)
TCP reno registered
NetWinder Floating Point Emulator V0.97 (double precision)
io scheduler noop registered
io scheduler anticipatory registered (default)
io scheduler deadline registered
io scheduler cfq registered
S3C_LCD clock got enabled :: 111.000 Mhz
LCD TYPE :: LTE480WV will be initialized
Window[0] - FB1: map_video_memory: clear ff200000:000bb800
```

```
FB1: map_video_memory: dma=53900000 cpu=ff200000 size=000bb800
Console: switching to colour frame buffer device 100x30
fb0: s3cfb frame buffer device
s3c_lcd_set_power:1
PWM channel 1 set g_tcnt = 1000, g_tcmp = 480
S3C ADC driver, (c) 2007 Samsung Electronics
S3C ADC driver successfully probed !
s3c-uart.0: s3c_serial0 at MMIO 0x7f005000 (irq = 37) is a S3C
s3c-uart.1: s3c_serial1 at MMIO 0x7f005400 (irq = 38) is a S3C
s3c-uart.2: s3c_serial2 at MMIO 0x7f005800 (irq = 39) is a S3C
s3c-uart.3: s3c_serial3 at MMIO 0x7f005c00 (irq = 40) is a S3C
RAMDISK driver initialized: 16 RAM disks of 4096K size 1024 blocksize
loop: module loaded
Driver 'sd' needs updating - please use bus_type methods
S3C NAND Driver, (c) 2007 Samsung Electronics
S3C NAND Driver is using hardware ECC.
NAND device: Manufacturer ID: 0xec, Chip ID: 0xdc (Samsung NAND 512MiB 3,3V 8-bit)
Creating 5 MTD partitions on "NAND 512MiB 3,3V 8-bit":
0x00000000-0x00040000 : "Bootloader"
0x00040000-0x00340000 : "Kernel"
0x00340000-0x00d40000 : "resource"
0x00d40000-0x11140000 : "File System 1"
0x11140000-0x20000000 : "User space"
UBI: attaching mtd3 to ubi0
UBI: physical eraseblock size: 262144 bytes (256 KiB)
UBI: logical eraseblock size: 258048 bytes
UBI: smallest flash I/O unit: 2048
UBI: VID header offset: 2048 (aligned 2048)
UBI: data offset: 4096
UBI: attached mtd3 to ubi0
UBI: MTD device name: "File System 1"
UBI: MTD device size: 260 MiB
UBI: number of good PEBs: 1040
UBI: number of bad PEBs: 0
UBI: max. allowed volumes: 128
UBI: wear-leveling threshold: 4096
UBI: number of internal volumes: 1
UBI: number of user volumes: 1
UBI: available PEBs: 10
UBI: total number of reserved PEBs: 1030
UBI: number of PEBs reserved for bad PEB handling: 10
UBI: max/mean erase counter: 3/1
ohci_hcd: 2006 August 04 USB 1.1 'Open' Host Controller (OHCI) Driver
UBI: background thread "ubi_bgt0d" started, PID 223
usb_host_clk_en
otg_phy_init ..................!
s3c2410-ohci s3c2410-ohci: S3C OHCI
s3c2410-ohci s3c2410-ohci: new USB bus registered, assigned bus number 1
s3c2410-ohci s3c2410-ohci: irq 47, io mem 0x74300000
usb usb1: configuration #1 chosen from 1 choice
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 1 port detected
s3c6410_OTGHCD s3c6410_OTGHCD: S3C6410 OTGHCD
s3c6410_OTGHCD s3c6410_OTGHCD: new USB bus registered, assigned bus number 2
s3c6410_OTGHCD s3c6410_OTGHCD: irq 58, io mem 0x7c000000
===============Enter ID_DEVICE Mod===============
===============Enter ID_DEVICE Mod===============
usb usb2: configuration #1 chosen from 1 choice
hub 2-0:1.0: USB hub found
hub 2-0:1.0: 1 port detected
otg_phy_init ..................!
===============Enter ID_DEVICE Mod===============
===============Enter ID_DEVICE Mod===============
otg_phy_init ..................!
Loaded s3c-udc version Aug 23 2010 (DMA Mode)
input: midfun-keys as /devices/platform/midfun-keys/input/input0
input: s3c-keypad as /devices/platform/s3c-keypad/input/input1
s3c-keypad Initialized
S3C Keypad Driver
S3C24XX RTC, (c) 2004,2006 Simtec Electronics
s3c2410-rtc s3c2410-rtc: rtc disabled, re-enabling
s3c2410-rtc s3c2410-rtc: rtc core: registered s3c as rtc0
```

```
S3C PWM Driver, (c) 2006-2007 Samsung Electronics
i2c /dev entries driver
s3c2410-i2c s3c2410-i2c: slave address 0x10
s3c2410-i2c s3c2410-i2c: bus frequency set to 108 KHz
s3c2410-i2c s3c2410-i2c: i2c-0: S3C I2C adapter
s3c-hsmmc: card inserted.
[s3c_hsmmc_probe]: s3c-hsmmc.1: at 0xc5000000 with irq 57. clk src: sclk_DOUTmpll_mmc1
TCP cubic registered
NET: Registered protocol family 1
NET: Registered protocol family 17
NET: Registered protocol family 15
ieee80211: 802.11 data/management/control stack, git-1.1.13
ieee80211: Copyright (C) 2004-2005 Intel Corporation <jketreno@linux.intel.com>
ieee80211_crypt: registered algorithm 'NULL'
ieee80211_crypt: registered algorithm 'WEP'
ieee80211_crypt: registered algorithm 'CCMP'
ieee80211_crypt: registered algorithm 'TKIP'
VFP support v0.3: implementor 41 architecture 1 part 20 variant b rev 5
s3c2410-rtc s3c2410-rtc: setting system clock to 2011-02-06 04:43:51 UTC (1296967431)
UBIFS: recovery needed
UBIFS: recovery completed
UBIFS: mounted UBI device 0, volume 0, name "rootfs"
UBIFS: file system size: 259854336 bytes (253764 KiB, 247 MiB, 1007 LEBs)
UBIFS: journal size: 12902400 bytes (12600 KiB, 12 MiB, 50 LEBs)
UBIFS: media format: w4/r0 (latest is w4/r0)
UBIFS: default compressor: lzo
UBIFS: reserved for root: 4952683 bytes (4836 KiB)
VFS: Mounted root (ubifs filesystem).
Freeing init memory: 104K
UART err int.
s3c-nand: 1 bit(s) error detected, corrected successfully
s3c-nand: 1 bit(s) error detected, corrected successfully
s3c-nand: 1 bit(s) error detected, corrected successfully
mmc0: host does not support reading read-only switch. assuming write-enable.
mmc0: new SD card at address ebe2
mmcblk0: mmc0:ebe2 SD512 500224KiB
mmcblk0: p1
s3c-nand: 1 bit(s) error detected, corrected successfully
s3c-nand: 1 bit(s) error detected, corrected successfully
s3c-nand: 1 bit(s) error detected, corrected successfully
UBI: attaching mtd4 to ubi1
UBI: physical eraseblock size: 262144 bytes (256 KiB)
UBI: logical eraseblock size: 258048 bytes
UBI: smallest flash I/O unit: 2048
UBI: VID header offset: 2048 (aligned 2048)
UBI: data offset: 4096
UBI: attached mtd4 to ubi1
UBI: MTD device name: "User space"
UBI: MTD device size: 238 MiB
UBI: number of good PEBs: 955
UBI: number of bad PEBs: 0
UBI: max. allowed volumes: 128
UBI: wear-leveling threshold: 4096
UBI: number of internal volumes: 1
UBI: number of user volumes: 1
UBI: available PEBs: 0
UBI: total number of reserved PEBs: 955
UBI: number of PEBs reserved for bad PEB handling: 9
UBI: max/mean erase counter: 2/1
UBI: background thread "ubi_bgt1d" started, PID 500
UBIFS: recovery needed
UBIFS: recovery completed
UBIFS: mounted UBI device 1, volume 0, name "ubifs1"
UBIFS: file system size: 240758784 bytes (235116 KiB, 229 MiB, 933 LEBs)
UBIFS: journal size: 12128256 bytes (11844 KiB, 11 MiB, 47 LEBs)
UBIFS: media format: w4/r0 (latest is w4/r0)
UBIFS: default compressor: lzo
UBIFS: reserved for root: 4952683 bytes (4836 KiB)
usbcore: registered new interface driver usbhid
drivers/hid/usbhid/hid-core.c: v2.6:USB HID core driver
s3c-nand: 1 bit(s) error detected, corrected successfully
s3c-nand: 1 bit(s) error detected, corrected successfully
```

```
Adding 64492k swap on /mnt/swap/actual_swap. Priority:-1 extents:1 across:64492k
usb 1-1: new full speed USB device using s3c2410-ohci and address 2
usb 1-1: not running at top speed; connect to a high speed hub
usb 1-1: configuration #1 chosen from 1 choice
================> wifi status change from 0 to 1
rtusb init --->


=== pAd = c4f01000, size = 439728 ===

RTMPAllocAdapterBlock, Status=0
usbcore: registered new interface driver rt2870
RTMPAllocTxRxRingMemory, Status=0
-->RTUSBVenderReset
RTUSBVenderReset
Key1Str is Invalid key length(0) or Type(0)
Key2Str is Invalid key length(0) or Type(0)
Key3Str is Invalid key length(0) or Type(0)
Key4Str is Invalid key length(0) or Type(0)
1. Phy Mode = 0
2. Phy Mode = 0
NVM is Efuse and its size =2d[2d0-2fc]
3. Phy Mode = 0
RTMPSetPhyMode: channel is out of range, use first channel=1
== rt28xx_init, Status=0
0x1300 = 00073200
s3c_lcd_set_power:1
PWM channel 1 set g_tcnt = 1000, g_tcmp = 96
Terminate the task(RtmpMlmeTask) with pid(559)!
Terminate the task(RtmpCmdQTask) with pid(560)!
Terminate the task(RtmpTimerTask) with pid(558)!
---> RTMPFreeTxRxRingMemory
- RTMPFreeTxRxRingMemory
================> wifi status change from 1 to 0
usb 1-1: USB disconnect, address 2
rtusb_disconnect: unregister usbnet usb-s3c24xx-1
RtmpOSNetDevDetach(): RtmpOSNetDeviceDetach(), dev->name=ra0!
RTUSB disconnect successfully
```

Categoty:Literati

Category:

- Development Boards

> From: eLinux.org

Lithium Ion Charger

M8022 Charger Schematic

> From: eLinux.org

Mini LA

Mini LA is a project of simple and cheap logic analyzer designed for amateur and semi-professional work.



Features:

- Up to 32 channels
- 128 Kb of memory for each channel
- Sampling rate up to 100MHz (timebase in 1-2-5 sequence)
- External clock input
- Input levels compatible with 3.3V and 5V logic
- Selectable pretrigger/posttrigger buffer size in 8K steps
- 16 bits wide trigger (0, 1, don't care)
- Programmable min. trigger-event width (1-16)
- Programmable trigger-events counter (1-16)
- External trigger input
- Communicating via LPT port (EPP mode support) or USB
- Documentation and source codes released under GNU GPL
- Mini LAParts

From: eLinux.org

Mobile Pro

This is the site for the NEC **Mobile Pro** 770/780/790 set of Handheld PCs. These handhelds are rather neat, and can make a powerful target for a strong linux system. Among the overall specs of these things we have:

- VR4121 168mhz 32/64 bit processor (PD30121) - you can access some info about it here: http://www.linuxdevices.com/products/PD3171456766.html
- 32mb of RAM memory
- 24mb of ROM memory
- 16mb of Flash (only on the 790 model)
- A PC card bridge serving one PCMCIA slot and one Compact Flash slot
- Near full size keyboard
- 640x240 64k color DSTN LCD screen with touch panel
- 56k modem
- IrDA
- NEC proprietary serial plug, requires NEC cable or just wire a cable yourself and solder it onto the pins
- VGA out

Components inside it listed and checked so far (this list was made while fixing a problem with the PCMCIA socket, so it will be updated later with more accurate information) - refer to the board picture for more info, I will try to get a scan of it later.

- NEC VR4121A D30121F1 (PD30121) Processor
- RICOH RFC5C396L PC Card controller
- CONEXANT RP56LD (Modem controller)
- Media Q MQ200 LCD Controller
- 2x HYUNDAI GM72V661641CLT7J 64mbit chips onboard
- 2x HYUNDAI GM72V661641CLT7J 64mbit chips on a plugable board (memory upgrade anyone?)
- 2x NEC 023C128040L (???) -> Supposedly the ROM chips, but I can't find any information. I haven't really looked for it yet either.

Here we got a pic of the thing: mp-open.jpg

The front - CF slot in the middle, headphones jack and microphone at the right: mp-front.jpg

PCMCIA Socket on the right side: mp-pcmcia.jpg

Back of the device (battery): mp-back.jpg

Bottom of the device, you can see the RAM/ROM plugable boards compartment's cover: mp-bottom.jpg

And the board picture mentioned before: mp-board.jpg

I'm currently working using the linux-mips.org VR41xx code with a few modifications and code obtained from Linux on MCR700 project. Development is going on continuously on this device, the page will be updated when deemed necessary.

Contact me for more info on: mendoza dot ricardo at gmail dot com

I would be interested in helping out on this project - how do i contact you? - Paul

Category:

- Hardware

> From: eLinux.org

MUSB

Contents

MUSBMHDRC DRD OTG Controller

The MUSBMHDRC DRD OTG Controller (from now on referred to as MUSB is a Dual-Role OTG IP Core used in several SoC implementations. At the time of this writing at least OMAP, DaVinci and Blackfins integrate that IP Core in the SoC and a discrete version of it is supplied by Texas Instruments as the tusb6010 ASIC.

MUSB PENDING PATCHES

Bug fix patches for v2.6.35

Greg's queue

Acked by Felipe

```
* None
```

Review in progress

Jon Povey: [1 patch] : Fixing compilation warning

```
* USB: musb: suppress warning about unused flags [1]
```

New feature patches for v2.6.36

Greg's queue

Anand: [1 patch] : Cleanup

```
* musb: Kill board specific pinmux from driver file [2]
```

Anand: [1 patch] : on Mentor DMA

```
* usb: musb: do not override DMA mode in channel program [3]
```

Acked by Felipe

```
* None
```

Review in progress

Sergei: [1 patch] : DA8x MUSB support

```
* MUSB: DA8xx/OMAP-L1x glue layer [4]
```

Ajay: [3 patch] : Set on AM35x MUSB support

```
* AM35x: Add musb support [5]
* musb: add musb support for AM35x [6]
* musb: AM35x: Workaround for fifo read issue [7]
```

Ajay: [2 patch] : FIFO table and cleanup

```
* usb: musb: Update FIFO mode_5_cfg to accomodate 4K [8]
* musb: remove extra blank and border lines [9]
```

Ajay: [3 patch] : Set on neednop flag for NOP

```
* OMAP3: musb: add neednop flag to fix nop modular issue [10]
* musb: populate board_data within musb structure [11]
* musb: use neednop flag for nop registration [12]
```

Ajay: [6 patch] : Set on SDMA as Mentor DMA workarounds

```
* musb: save OTG base physical address [13]
* musb: use system DMA to fix Inventra DMA issue on RTL-1.4 [14]
* musb: add function to check if Inventra DMA used [15]
* musb: use system DMA for unaligned buffers on RTL >= 1.8 [16]
* musb: gadget: fix tx transfer path for mode0 operation [17]
* musb: dma: use optimal transfer element for sdma [18]
```

Ajay: [1 patch] : On DMA channel release in host mode

```
* musb: host: release dma channels if no active io [19]
```

Hema: [2 patch] : on Mentor DMA

```
* usb: musb: Unmapping the dma buffer when switching to PIO mode [20]
* usb: musb: Dynamic dma channel allocation in gadget driver [21]
```

Hema: [4 patch] : Set on HWMODS

```
* usb: musb: Adding names for IRQs in resource structure [22]
* usb: musb: Remove board_data parameter from musb_platform_init() [23]
* usb: musb: HWMOD database structures addition for OMAP3 [24]
* usb : musb:USB driver using omap_device_build for device registration. [25]
```

Category:

- Hardware

> From: eLinux.org

Nand Flash256

Common 256megabit flash models:

- Media:k9f5608u0a.pdf Samsung
- Media:tc58256aft.pdf Toshiba
- Media:tc58dvm82a1ft.pdf Toshiba
- sdtnf-256 Sandisk(no datasheet but compatible)
- sdtngahe0-256 Sandisk(no datasheet but compatible)

Category:

- Hardware

> From: eLinux.org

Nor vs Nand

There is a good introduction on the differences between NOR and NAND flash at:

http://www.commsdesign.com/design_corner/OEG20031022S0011

Samsung seems to be able to boot off of NAND flash.

Category:

- Flash Memory

From: eLinux.org

NTSC Bitbang

Introduction

This document describes how to generate composite color video signals in software using an SX microcontroller. First the document describes the video signal and after that how to do it in software. There is also a PDF-Version of this document, that is better if you wnat to print it to paper. (Note: The PDF-file also contains the games pong and tetris with source, so you might not want to print the whole document, also note that the document is supposed to be printed on both sides of the paper making some of the even pages are empty)

Background

Back in early 1998 I made some experimenting using a PIC16F84 microcontroller (3MIPS of processor power) to generate composite B&W video signals on the fly in software, with two resistors as the only video hardware. I made the two classical games Pong and Tetris with this technique and published them including source on my homepage. Since then it has been built by several hundreds of people. During the Christmas 1998-1999 I got some equipment from Scenix (nowadays known as Ubicom) and made some experiments to generate color video signals using an SX chip, but before I got any results my programmer broke down, at least that was what I believed, and I stopped developing it. In the early summer of 2001 I was told by people at Parallax that it was the early versions of the SX-chips that had a bug in them so my programmer was just fine, so they gave me some new chips and I continued my work. After some new experiments, calculating and many late hours and a bit of luck I got my TV to lock onto the color signal an d by the end of summer I got a Tetris game up and running. During the fall I developed the Pong game, which was finished during the Christmas holidays 2001-2002. I didn’t release the games as there were some details left to take care of. I didn’t want to publish them until they were as perfect as possible due to my bad experience with my PIC-based games that were spread in early bad versions. Now in spring 2003 I decided that I shouldn’t do any more improvements of the games as I don’t have time to work on them and I got to stop sometime. The biggest remaining issue is that it only works good for NTSC, it is much harder to get a correct PAL signal in software, but that is a problem for someone else to solve. Another issue about the games was this text about generating color video signals that I wanted to finish before I released the games, to not get that many questions about video generation that I don’t have time to answer. After reading this document you will hopefully understand how to generate color composite video signals in software. To fully understand this you need mathematical knowledge at university level, some RF-knowledge would also help a lot.

Copyright note

> From: eLinux.org

Peek



Contents

The Device

In The News

- Company Website
- Linux Devices
- Geeky Peek Linux Challenge
- Amazon.com
- Peek teardown photos Maushammer on Flickr
- Discussion on irc.freenode.net channel #edev
- Specific discussion of Linux on the Peek on irc.freenode.net channel ##peek

Basic Specifications

- 320x240 TFT LCD with backlight
- 47 Key QWERTY keyboard
- 360 degree jogwheel with button
- Vibrator Motor
- Small Speaker
- Side Push Button
- Top Power Button
- Standard SIM Socket
- Micro-B USB Charging Socket (USB connectivity is **not** supported, this is really just a TTL Serial port and a charging socket.)
- Internal Coin Backup Battery
- E-Mail Notification LED
- Battery Door Contacts Switch

Discussions about the Current OS

- It's a lightweight, purpose-built OS called PeekUX. [1]
  - Starts with TI's very own version of the real-time OS called Nucleus (by Mentor)
  - TI has compiled in device drivers, and other stacks and turned it into their own platform.
  - Peek SW developers then took what TI gave them and wrap it with their own proprietary GUI and network libraries to create Peekux.
  - The email application then runs on top of that.
  - The whole thing is built as a single image. Thus the distinction between OS and App is blurred.
- "Hacking maybe difficult" article at Peek Forums. [2]

Wireless Network and FCC stuff

- Uses T-Mobile's nationwide GSM network.
- Also can use AT&T's GSM network as a secondary network.
- FCC ID is listed as: V6LPEEK0001
- Uses the GSM850 band. TX: 824-848MHz RX: 869-893MHz
- Maximum RF Radiated Power (GPRS850): 26dBm EIRP
- RX Sensitivity: -109dBm ?
- Antenna: PIFA type

Mail Handling

- Currently, the very large computing resources at Amazon AWS (Seattle, WA) is being used to handle the mail polling and forwarding for the Peek users. The mail handler was worked out at Peek HQ and then unleashed at AWS. AWS's

"EC2 Service" is immensely flexible and allows Peek to keep up with the growing user base.

Chip List

- Spansion 71NS128NB0BJWRN 32Mb Psram + 128Mb NOR Flash datasheet Additional flash info
- TI D6591BQA - TCS2310
  - The original TI link is dead, but there's a snapshot on the Internet Archive
- TI T3031FZH - TWL3031 power management and IO control
- RFMD RF7115 Quad Band GSM850/GSM900/DCS/PCS TRANSMIT MODULE datasheet
- ST STMPE2401 - Port expander with Keypad and PWM controller Datasheet
- 26.00 MHz XTAL for the TCS2310
- 32.768 kHz XTAL for the TWL3031

Battery

- Model: PK-BAT-001
- Voltage: 3.7V
- Capacity: 700mAh
- Maximum Charge Voltage:4.2V
- Standard: GB/T18287-2000

Peek Accessories

- Upgrade Cable for upgrade/hacking (No longer available).
- Replacement Battery (No longer available).

Opening the Case

- The case has 4 tiny "torx" style screws. The size is T6. You will need a Precision Torx Screwdriver set.
- Obviously, opening your Peek **voids the warranty**.
- There are 3 tiny screws immediately visible when you take off battery cover. And the fourth screw is hidden under the red "Do not remove" sticker.
- After the screws are removed, start from bottom of unit and carefully pry the top and bottom housings apart.
- To understand what I mean, see this picture from Maushammer on Flickr. On the left side of image is the bottom housing. Notice the four screw holes. The rest of the housing attaches with tabs that easily pry apart.

Debugging/Programming Tools

- Micro to Mini USB Adapter (does not convert the UART to USB)
- Flyswatter USB JTAG Debugger for upgrade/hacking
- OpenOCD JTAG Debugger Software -- currently doenst know anything about the LoCosto chips.
- UART TTL to USB Adapter for upgrade/hacking
- J-Link USB JTAG/SWD debugger from IAR Systems. User's Guide
- Same J-Link product from Segger USA
- Cheap but decent Benchtop DC Power Supply from Extech Instruments. This will save you from needing to keep charged batteries around.

External MicroUSB Charge/Upgrade port

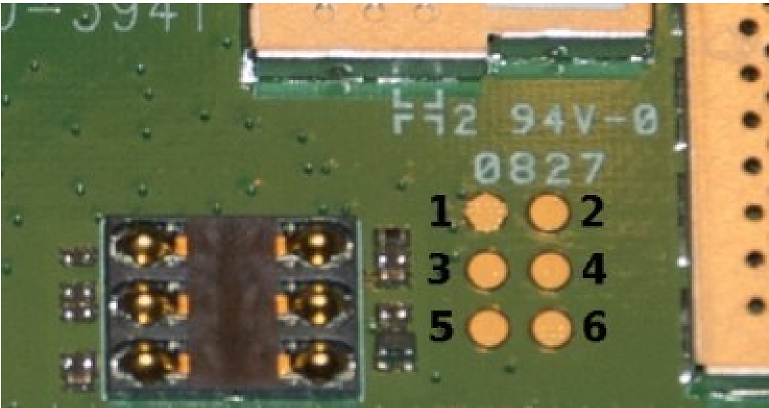
This is the MicroUSB port that is exposed on the left side of the peek. It's used for both charging and upgrading the Peek. The upgrade cable uses a serial TTL to USB level shifter to convert the serial TTL into a USB com port. a bootlog can be viewed using serial settings 115200,n,8,1

Pin	Function
1	+5V(USB Standard)
2	TXD +1.8V TTL
3	RXD +1.8V TTL
4	N/C
5	GND
Shield	Drain wire

Internal UART Header

This is on the inside of the Peek under the SIM (this is also covered by the warranty sticker). a bootlog can be viewed using serial settings 115200,n,8,1 **Note:** This is not a new, hidden serial port. This port is electrically the same as the External Charger/Upgrade port shown above.

Pin	Function
1	Ground
2	nPowerButton (active low)
3	UART_RX +1.8V TTL (input to Peek)
4	USB_PWR +5V
5	UART_TX +1.8V TTL (output from Peek)
6	Bat+ 4.2V

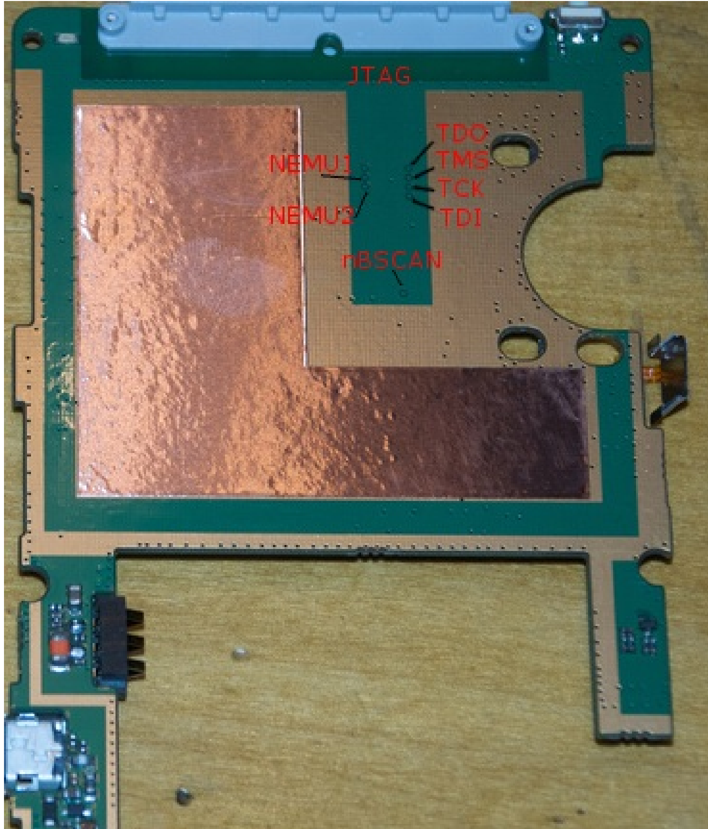


Picture from Maushammer on Flickr

JTAG Pads (vias)

These vias are found on the bottom side of printed circuit board just below the white antenna strip. ***Note: You will need to remove the solder mask and solder small wire to the vias. 26AWG or 30AWG is best.** Obviously, the four signals TDO, TMS, TCK, and TDI are the most important ones. Connect these signals straight to your JTAG dongle. You should also pick up a Ground signal somewhere near the DC Power circuity on the lower left section of PCB. Perhaps, depending on your JTAG dongle, you could also power the Peek from the dongle itself. ;-)

up a Ground signal somewhere near the DC Power circuity on the lower left section of PCB. Perhaps, depending on your JTAG dongle, you could also power the Peek from the dongle itself. ;-)



Thanks to "codeman" and "prpplague" for this (these pinouts may be wrong!).

uClinux

Bootloader

- a bootloader, such as U-Boot or APEX, will be needed to facilitate booting uClinux
- low level access to Flash memory via JTAG is now doable. (see above)
- U-Boot could be used, but is pretty large
- APEX is small, configurable, and easily adapted for ARM7TDMI

Kernel

- mainline kernel has the basic support for ARM7TDMI core
- need basic support for the TCS2310
- STMPE2401 keypad driver for linux-2.6 git (20090923) posted by Cfriedt
- stmpe24xx project page on SourceForge.net

Root Filesystem

- use Buildroot for initial testing
- use uClibc as core C library
- use BusyBox as basic utilities for testing

Tools

- use CSST (Cellular Systems Software Tools) from TI for loading Flash with bootloader etc. This is the Release Notes for the CSST included in SDP2430 (OMAP2430). There is a CSST for the TI LoCosto chips included with the Peek

The Peek update program calls a command line CSST app to actually load the new firmware to flash. In my case, the command line was:

```
D:\apps\Peek\CSST\csstcli.exe -t 1 -op download -i D:\apps\Peek\software\Rel_01_09_10.m0 -dt NOR -p COM5
```

This command can be executed from a command prompt. You will be prompted to reset the Peek (remove and replace the battery, then hit the power button) and the download will begin!

- use TMSH to execute commands on the Peek itself (source):

Your Peek must remain active during the processing of each command so you may need to move the scroll wheel or press a button intermittently during the process to keep it on. You must also have the Peek cable and Peek upgrade software installed on your computer before you start.

Power on Peek and connect Peek to PC using the Peek Upgrade cable. Determine wich port the Peek is connected to. Either Check Device Manager - Start > Run > devmgmt.msc OR right click My Computer > Properties > Hardware Tab and Look at LPT&COMM, expand + sign and you should see what COMM port your cable is on

Open a command prompt. If you plan on copying files to the computer you must right click "Command prompt" and run it as an administrator even if you are already logged on as an administrator. It should be located under "Accessories" in the start menu.

Type in:

```
cd "C:\Program Files (x86)\Peek.1.10.00\tmsh" (or whatever your Peek directory is called)
```

Then type:

```
tmsh -psX -ttlocosto
```

Where X = your COMM port # from earlier.

Here's a list of TMSH commands:

Command	Description
help	Print help for the command name on stdout Arguments
run	Run the script file named filename. Arguments
use	Load the ETM module represented by the DLL “name.dll” Arguments
ping	ping target Arguments [options]
quit	Quit the shell
auw	Write audio parameters
aur	Read audio parameters
aul	Load audio parameters from FFS file
aul	Save audio parameters to FFS file
mkfs	Format flash
fwr	File write from PC to target
frd	File read from target to PC
cd	Change current working directory
mkdir	Make a new directory
rm	Delete a file or directory

rm	Delete a file or directory
mv	Move or rename a file
pwd	Print working directory
ls	List files in directory
cat	View of text files
ln	Make a link between files
chmod	Change files/directories permissions
touch	Create an empty file
df	Show the amount of free space
fsq	Query ffs parameters and values
rfe	Enable RF – transmit and receive
se	Enable special customer task
scw	Setup the statistics configuration
scr	Read the statistics configuration
sr	Get statistics results
rfpw	Write RF parameters for receive mode
rfpr	Read RF parameters for receive mode
rxpw	Write RX parameter for receive mode
rxpr	Read RX parameters for receive mode
txpw	Set parameters for transmit mode
txpr	Read parameters for transmit mode
spw	Write customer special parameter
spr	Read customer special parameter
rftw	Write a RF table
rftr	Read a RF table
ttw	Write a ramp template to transmitter
ttr	Read a ramp template from transmitter
stw	Write a customer special table
str	Read a customer special table
mr	Read value from memory address
mw	Write value to memory address
cr	Read value from codec register
cw	Write value to codec register
me	Enable miscellaneous parameter
mpr	Read miscellaneous parameter
mpw	Write miscellaneous parameter
vg	Get hardware or software version

Email is kept in a database. Location Peek/peek.db Attachments are kept in location Peek/Attach The attachment will be named with a string of numbers_filename.extension_more numbers example 9585785_yourpic.jpg_552694 Pictures will be re-sized by Peek servers before they are delivered to your Peek. As an example a picture I sent to my Peek started at 998x1274 the picture downloaded from my Peek was only 152x195.

References

Other Peek Hacking Sites

- The official blog started by Peek's founder to help people get started in hacking the Peek
- PeekLinux.com

Core Chipset Tools and Diagrams

- ZIP file released by Peek containing the files you need

Similar board bringups

- OMAP2420 with U-Boot

TI documents

- TI Wireless Solutions Guide. the TCS2310 appears on Page 17.

Watch a Peek being assembled

- Watch Gabe Fabius and Dan Morel of PEEK, Inc. discuss the inner guts of the Peek device.

Category:

- Hardware Hacking

> From: eLinux.org

Pixter

(picture from [amazon] where you can [order] it for $79.99)





The Pixter is like a personal digital assistant (PDA) for kids. But instead of storing boring appointments and phone numbers, they get to draw, do puzzles, and learn a little, too. The Pixter is about the size of a small book, and has a pretty solid weight to it. It's made out of brightly colored and durable plastic. The main action of this toy is drawing with an attached stylus. The Color Pixter offers much better graphics then the black and white version, which comes in handy with activities like Paint-by-Numbers and Connect-the-Dots. You can draw a picture free form, or you can use one of the starter backgrounds. There are tool buttons, like on a PDA, that give you access to fun stamps, special effects, and the eraser button. You can also save your creation in the Pixter's memory. There are some boppy tunes playing along with all this

creative activity, but luckily for parents, there's also a jack for headphones and a volume adjuster. The Pixter Expansion Slot on top of the Pixter accepts ROM-based expansion cards that contain additional software and games. Requires 4 'AA' batteries. (Not included.)

The LH75411 controller was developed to be a perfect fit to the Pixter's hardware and memory needs. Based on a 70-MHz ARM7TDMI core, the chip includes color LCD controller circuitry; a 10-bit, eight-channel A/D converter, an integral touchscreen controller; and the usual assortment of counter/timers, programmed I/O pins and SRAM. Pivotal to the design were the the 32 kbytes of SRAM, which make it possible to execute the Pixter's algorithms locally, and outputs that can be configured to implement a 12-bit pulse-width-modulated audio output. The addition of an inexpensive external capacitor gave the Pixter the functionality of a moderate-quality audio D/A converter.

There are several projects in the works to make an external cartridge to boot a small Embedded OS. conversaion on hacking the Pixter can be found on irc.freenode.net #pixterdev and #eLinux

- Pixter Chip List
- Pixter Expansion Slot
- RMS100 - similar hardware
- Pixter Dev Board
- PixterMMC
- Pixter Camera
- Pixter Dev Cart

Category:

- Pixter

From: eLinux.org

Pixter Multimedia



Items of interest inside the device:

- Sharp LH79524 CPU
- Two IC42S16100-7T SDRAM ICs (2MB each, 4MB total) [datasheet]
- TLV320DAC26 audio [datasheet]
- OTP flash Chip On Board IC
- SPI eeprom Chip On Board IC (still collecting information on this)
- Audio Chip On Board IC which connects to the DAC26's microphone input (still collecting information on this)
- Well-labeled PixterMultimediaLCD port
- 4-wire touch-screen
- Silkcreen above largest Chip On Board reads "PT1543A-BGA-42FC 2005/04/30 REC 4.2B"
- Pixter Multimedia Expansion Slot
- Pixter Multimedia Expansion Cartridge
- Pixter Multimedia Developer Board
- PixterMultimediaJTAG

PCB Pads:

Primary side:

- TXD0, RXD0 these are incorrectly label on the pcb, they are actually UARTTXD1 and UARTRXD1
- CTS0, RTS0 (apparently extended to cartridge port)
- L16 (nCS0 PM0)
- A4 (PH5/ETHERTX1 I/O General Purpose I/O Signals -- Port H5; multiplexed with Ethernet Transmit Channel 1)

USB Pins:

- A16 USBDN I/O USB Data Negative (Differential Pair output, single ended and Differential pair input)
- A15 USBDP I/O USB Data Positive (Differential Pair output, single ended and Differential pair input)

Secondary side:

- Test1
- Vss
- Test2
- CS1 bridge (SMT pads for 0-ohm resistor with a line between them) opposite the big CoB blob
- M4 (PB4/SSPRX/I2SRXD/UARTRX0/UARTIRRX0) Port B4; multiplexed with SSP Data In, I2S Data In, UART0 Serial Data In, and UART0 Infrared Data In
- 1V8 (most likely 1.8v for the cpu core)

Category:

- Pixter

> From: eLinux.org

Programmers Hardware Database

The Programmers Hardware Database is a wiki project aimed at collecting hardware information that is mostly relevant for software developers, including, but not limited to:

- links to datasheets / user's manuals / technical reference manuals
- memory maps / register layouts (when datasheets are not available)
- sample code
- links to software projects
- ...

This information is collected for various devices, such as:

- ICs (CPUs, SoC, application processors...)
- consumer electronics
- PC hardware
- peripheral hardware
- embedded development systems
- ...

Currently the most active category of topics is embedded hardware, most notably consumer electronics, embedded development systems and the SoCs used in these devices. Please have a look at the recent changes for an up to date view of new wiki pages.

For a full motivation and description of project goals, have a look at the about page.

Please feel free to make use of and link to the wiki pages of the PHD at your desire. Also, feel free to contribute any kind of hardware information to the PHD. Any suggestions and input from the Embedded Linux community, and an exchange of knowledge and expertise, are highly appreciated.

> From: eLinux.org

R8610 Based WAP

From Linuxdevices.com article

Tiny WAPs (Wireless Access Points) share files and printers - $115 Jun. 21, 2006

http://linuxdevices.com/files/misc/ovislink_wmu6000-drive.jpg

Half a dozen companies around the world are shipping tiny Linux-based wireless access points (WAPs) with built-in file and printer servers. The WAPs appear to run a 2.4-series Linux kernel ... The design appears to be based on RDC Semiconductor's R8610, a 133MHz RISC-based SoC that executes the i486 instruction set at about 44 bogoMIPS.

... 16KB of L1 cache, a 33MHz PCI bus, an external SDRAM/ROM/memory controller, IPC (internal peripheral controller) with DMA and IRQ timer/counter, a 10/100 Ethernet MAC, FIFO UART, and USB 2.0 host controller. ... 32MB of SDRAM, 4MB of Flash, and support user-supplied 2.5-inch (laptop-sized) hard drives formated with FAT, FAT32, or ext2. Claimed wireless throughputs up to 6MB/s ...

... an 802.11G WiFi interface supporting WPA and WEP encryption. The interface can be configured as a wireless client, access point, WDS bridge, or WDS repeater ...

Dual USB 2.0 ports can be used to connect memory card readers ... The ports also support USB OTG ("on the go") [and] LPR print server software for USB printers.

Designs believed to be based on the RDC SoC include:

- OvisLink WMU-6000FS
- Micronica MGB100
- SafeCom SWSAPUR-5400
- Level One WAP-0007
- SMC WAPS-G EU
- CometLabs MGB-100

Availability

WAPs based on the RDC R8610 appear to be available now, priced at about $115.

Category:

- Hardware Hacking

> From: eLinux.org

Reciva Barracuda

The Barracuda Module is a Samsung S3C2410 based cpu module which is used in some of the Reciva internet radio devices.

Module Datasheet

Category:

- Hardware

From: eLinux.org

SM501-User Level Device Driver

Contents

概要

割り込みフックドライバ

Media:irqhook-0.0.4.tar.gz

サンプルアプリケーション

SM501 UARTシリアルターミナル for RTS7715R2D

Media:sm501uart_tty-0.0.1.tar.gz

実験

目的

割り込み処理をユーザタスクで行う場合に問題となる、割り込み応答時間のレイテンシ増分を測定する。

環境

- Target platform

Renesas RTS7751R2D

SH4(SH7751) 240MHz

64MB SDRAM

10Base-T Ethernet

Linux 2.6.16.4 / 2.4.20

256MB Compact Flash for rootfs

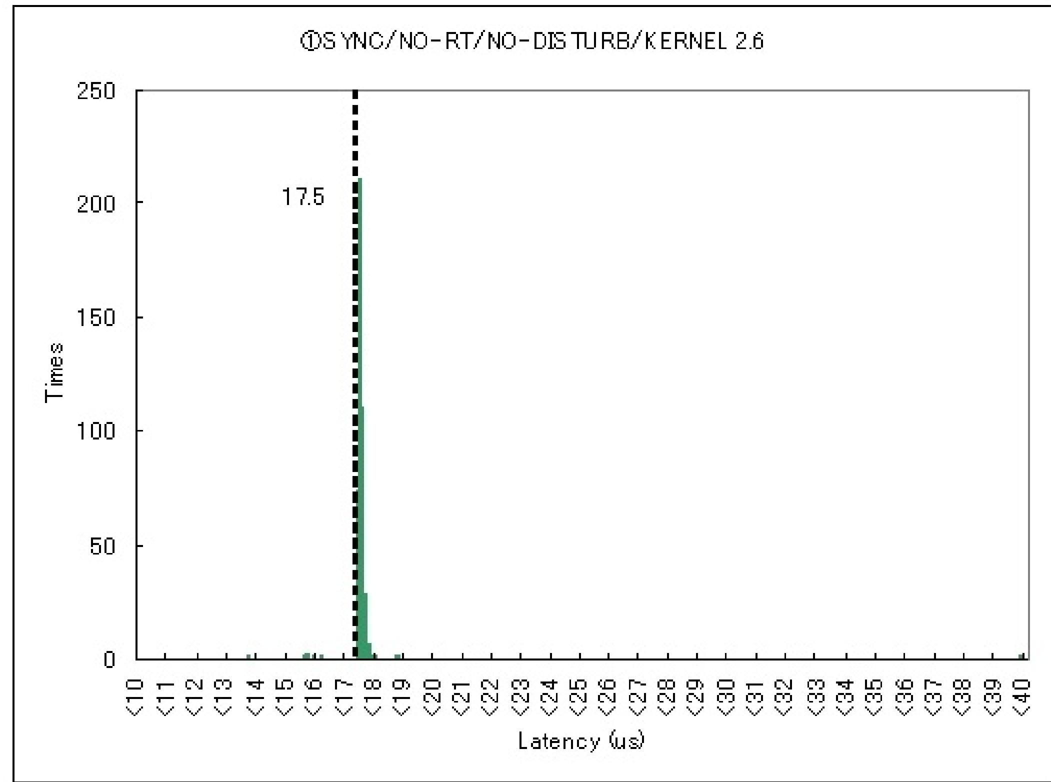
方法

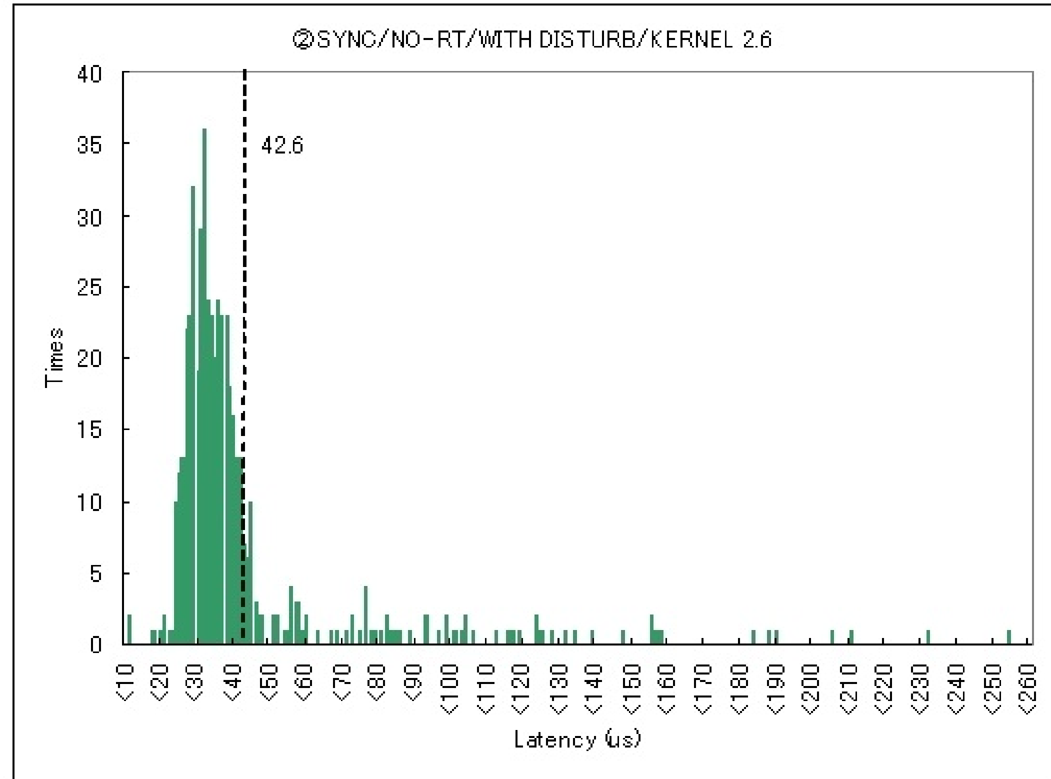
シリアルからデータを送信し、割り込み発生後、カーネルドライバの割り込みハンドラが起動してから、ハンドラによってwakeup()されたULDDが起床するまでの時間を測定する。

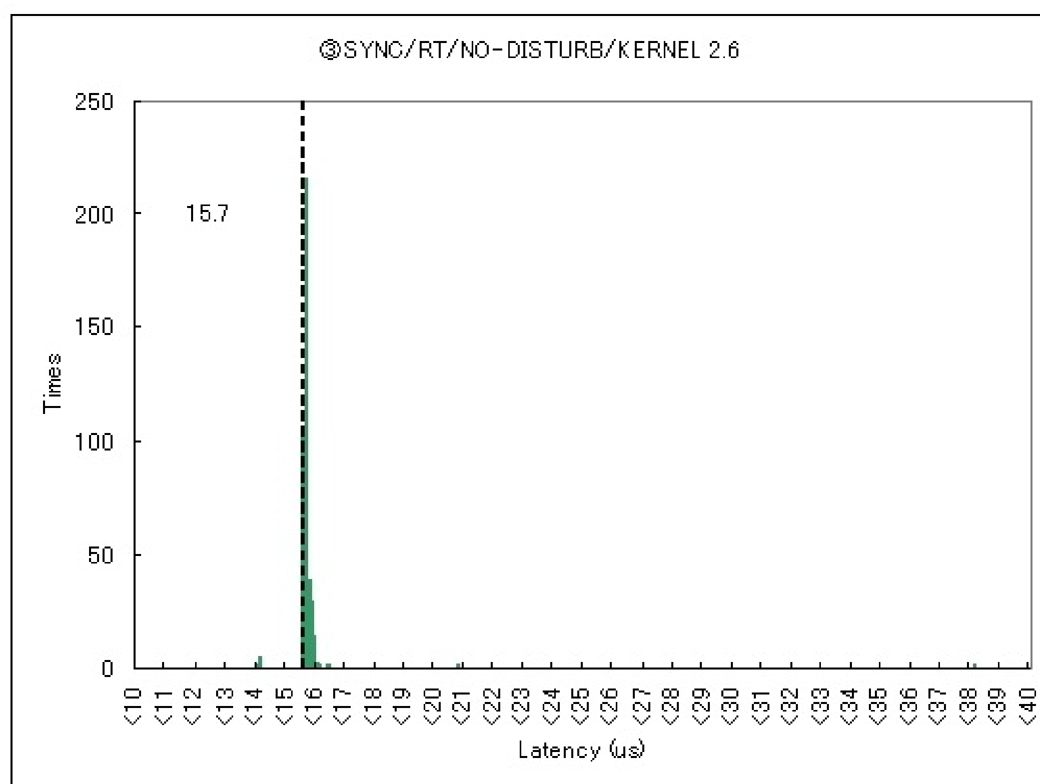
- 10KBのデータを38400bpsで送信
- データ受信時の起床時間（カーネル割り込みハンドラ起動直後からULDD復帰までの時間を計測）を計測
- 負荷状況を変化
    - (none): 負荷なし
    - lat_proc: lmbench3付属のプロセス生成ベンチマークをバッググラウンドで実行(./lat_proc -P 3 -N 20 exec &)
- Non-RT(SCHED_OTHER), RT(SCHED_RR, sched_prio=1)スレッドを使用

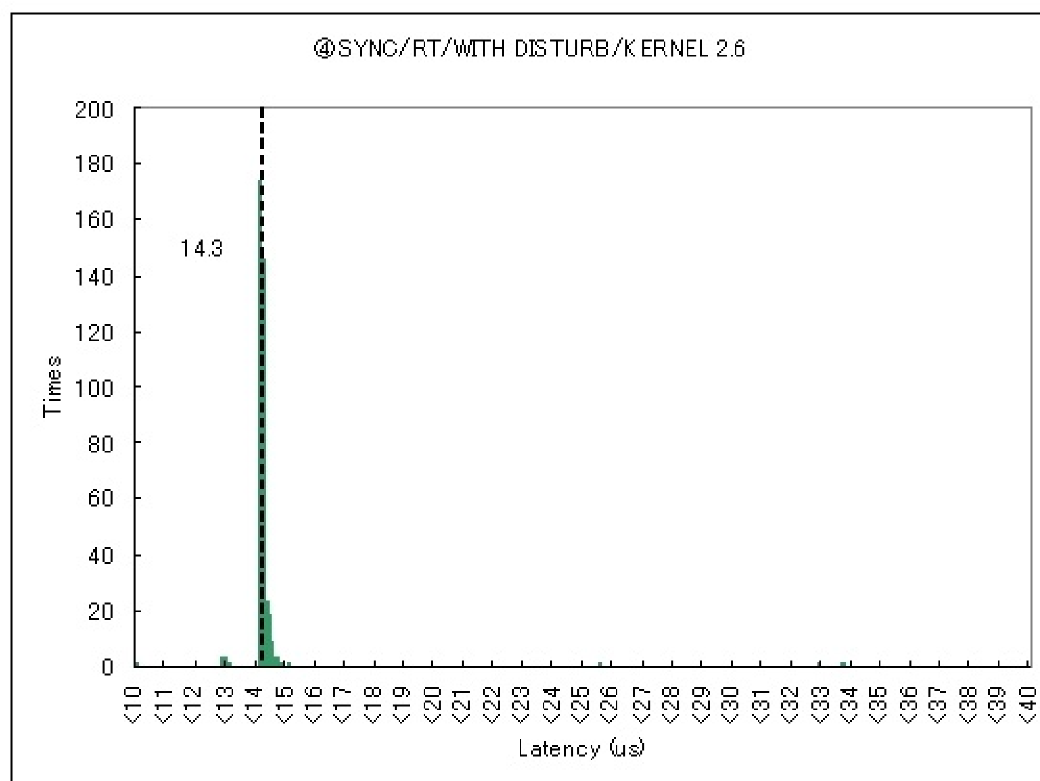
結果

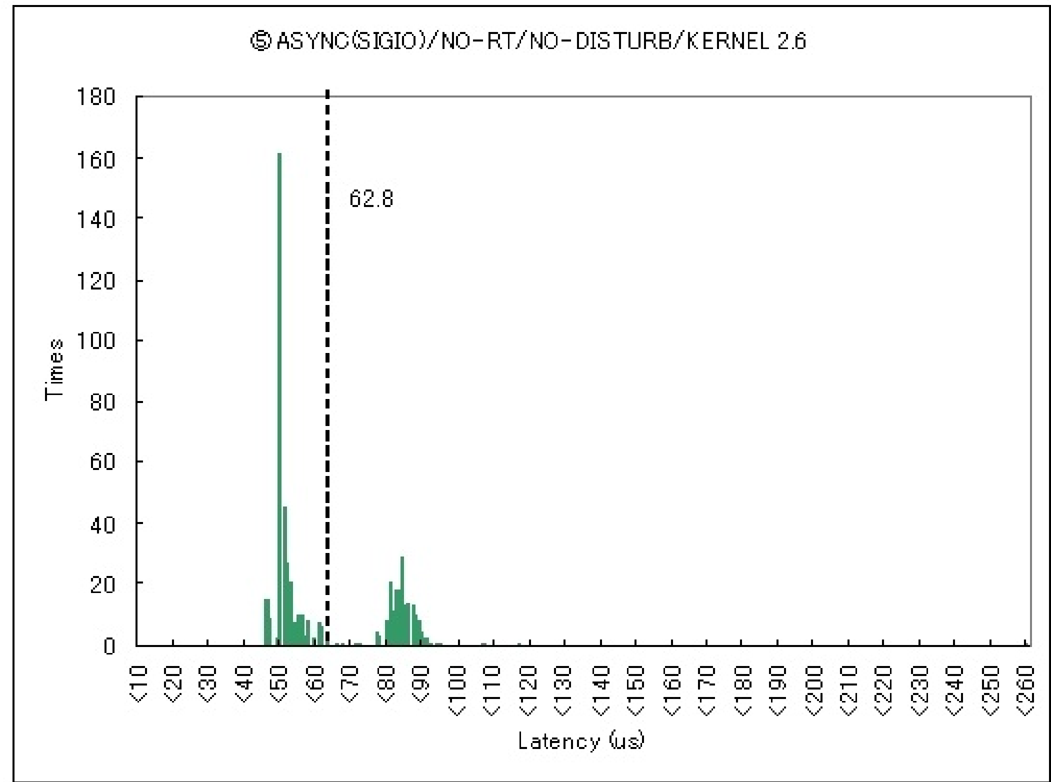
2" | **IRQ Delivering Method**	2" | **Scheduling Policy**	2" | **Disturbance Task**	**Linux 2.6.16.4 (CONFIG_PREEMPT=y)**	**Linux 2.4.20**		
MAX (us)	**AVG (us)**	**MIN (us)**	**Plot**	**MAX (us)**	**AVG (us)**	**MIN (us)**
File I/O (sync)	SCHED_OTHER	(none)	39.80	17.52	13.73	①
SCHED_OTHER	*lat_proc*	253.27	42.57	10.93	②	2808.
SCHED_RR	(none)	38.13	15.73	14.07	③	17.
SCHED_RR	*lat_proc*	33.73	14.32	8.80	④	2697.
SIGIO (async)	SCHED_OTHER	(none)	524.00	62.84	45.47	⑤
SCHED_OTHER	*lat_proc*	72317.40	2185.96	83.33	⑥	
SCHED_RR	(none)	81.40	47.12	43.67	⑦	
SCHED_RR	*lat_proc*	280.93	102.89	79.40	⑧	
SIGRT (async)	SCHED_OTHER	(none)	83.47	50.00	44.80	⑨
SCHED_OTHER	*lat_proc*	85973.93	2528.19	85.40	⑩	
SCHED_RR	(none)	75.93	46.71	43.07	⑪	
SCHED_RR	*lat_proc*	290.80	102.18	78.87	⑫	

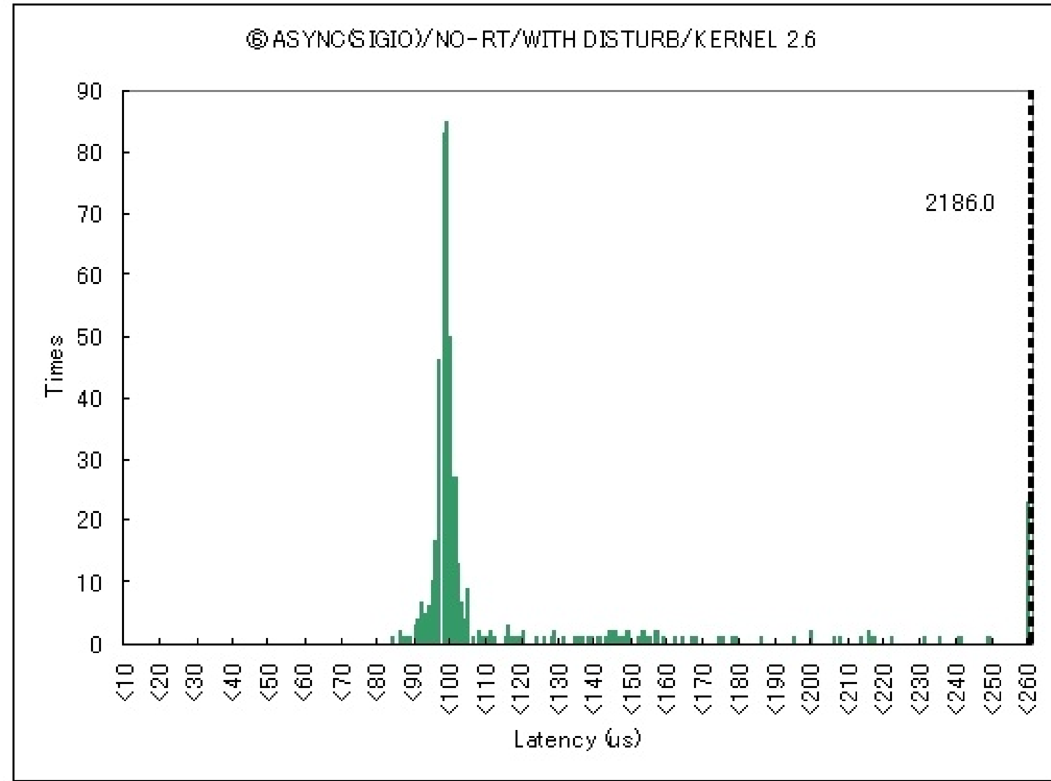

①SYNC/NO-RT/NO-DISTURB/KERNEL 2.6
250
200
150
100
50
0
Times
17.5
<10 <11 <12 <13 <14 <15 <16 <17 <18 <19 <20 <21 <22 <23 <24 <25 <26 <27 <28 <29 <30 <31 <32 <33 <34 <35 <36 <37 <38 <39 <40
Latency (us)

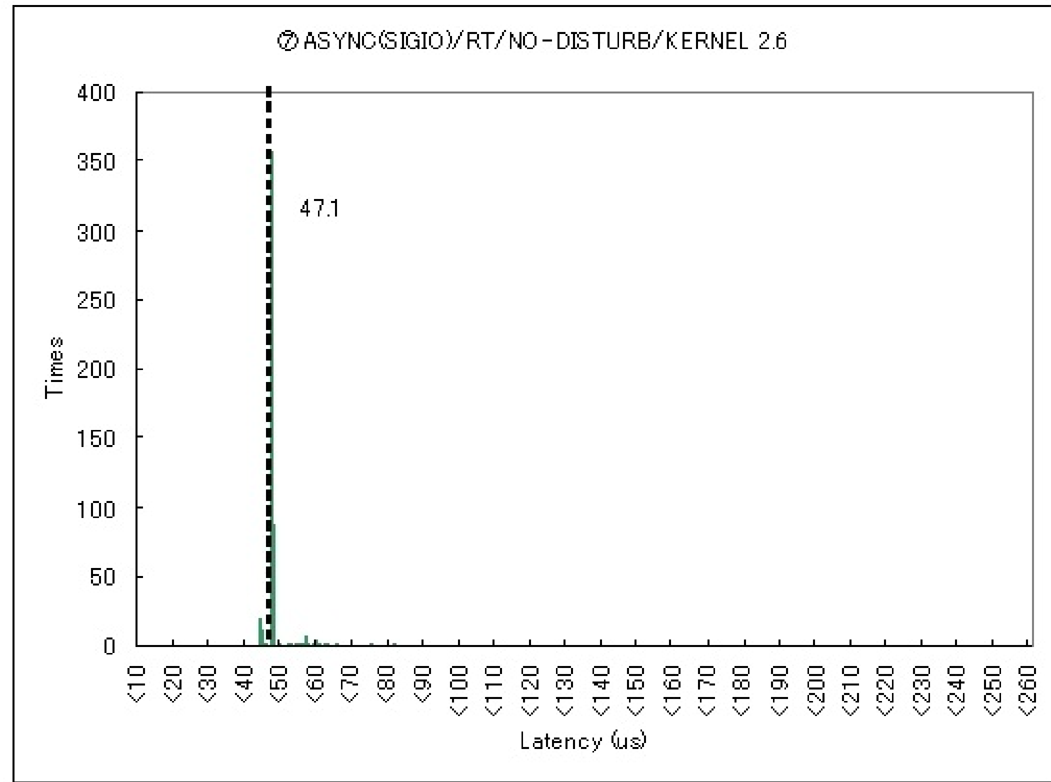

②SYNC/NO-RT/WITH DISTURB/KERNEL 2.6
40
35
30
25
20
15
10
5
0
Times
42.6
<10 <20 <30 <40 <50 <60 <70 <80 <90 <100 <110 <120 <130 <140 <150 <160 <170 <180 <190 <200 <210 <220 <230 <240 <250 <260
Latency (us)

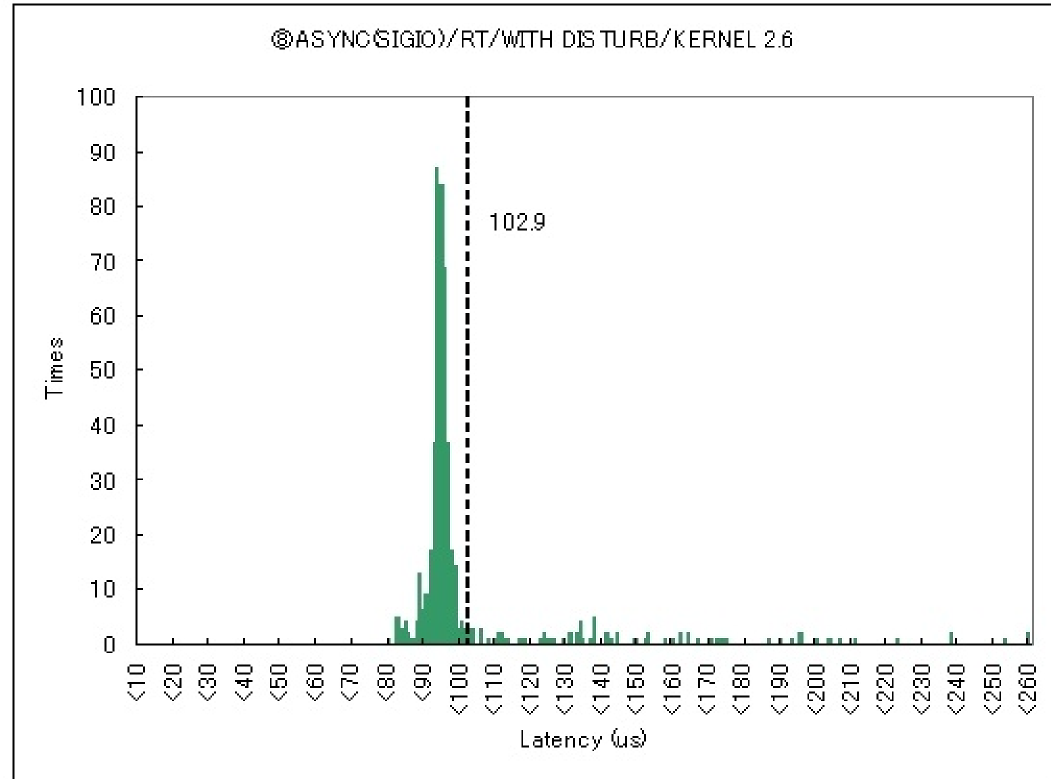

③SYNC/RT/NO-DISTURB/KERNEL 2.6
Times
250
200
150
100
50
0
15.7
<10 <11 <12 <13 <14 <15 <16 <17 <18 <19 <20 <21 <22 <23 <24 <25 <26 <27 <28 <29 <30 <31 <32 <33 <34 <35 <36 <37 <38 <39 <40
Latency (us)

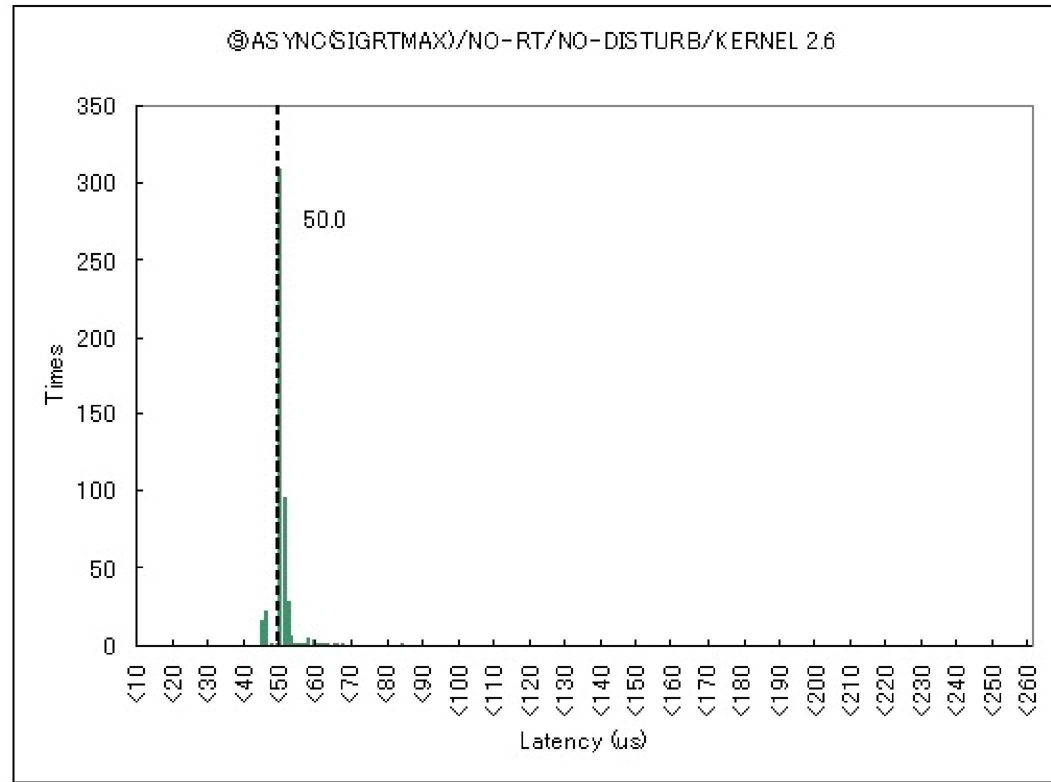

④SYNC/RT/WITH DISTURB/KERNEL 2.6
Times
200
180
160
140
120
100
80
60
40
20
0
14.3
<10 <11 <12 <13 <14 <15 <16 <17 <18 <19 <20 <21 <22 <23 <24 <25 <26 <27 <28 <29 <30 <31 <32 <33 <34 <35 <36 <37 <38 <39 <40
Latency (us)

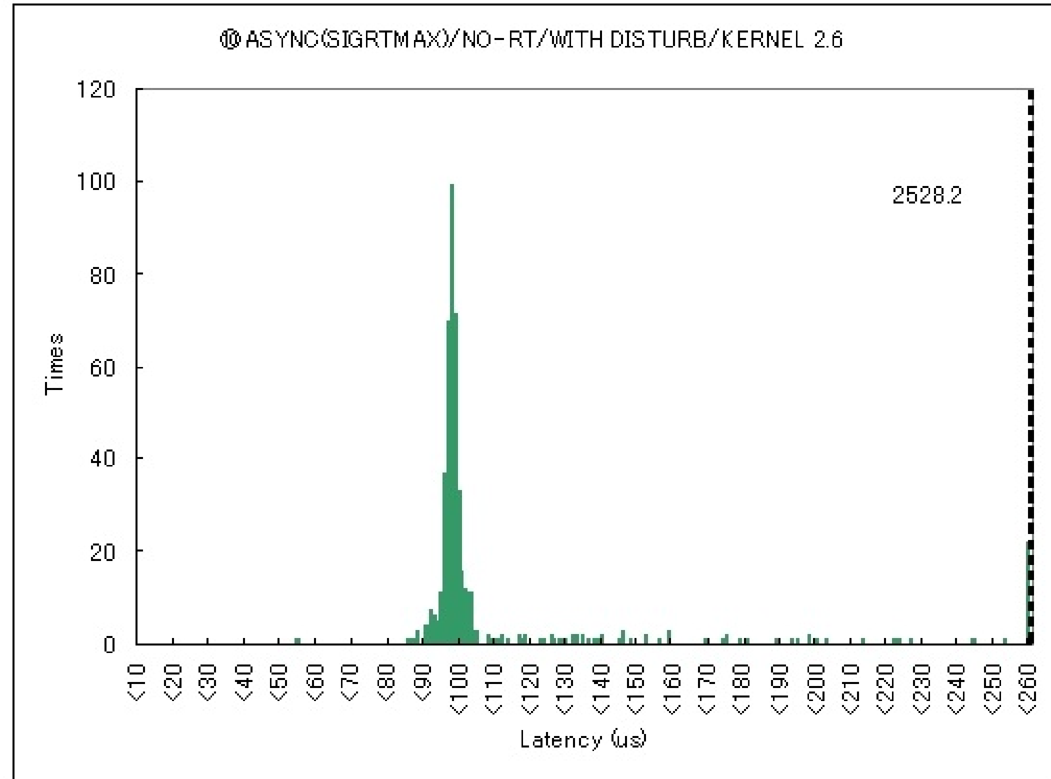

⑤ASYNC(SIGIO)/NO-RT/NO-DISTURB/KERNEL 2.6
62.8
Times
Latency (us)
180
160
140
120
100
80
60
40
20
0
<10
<20
<30
<40
<50
<60
<70
<80
<90
<100
<110
<120
<130
<140
<150
<160
<170
<180
<190
<200
<210
<220
<230
<240
<250
<260

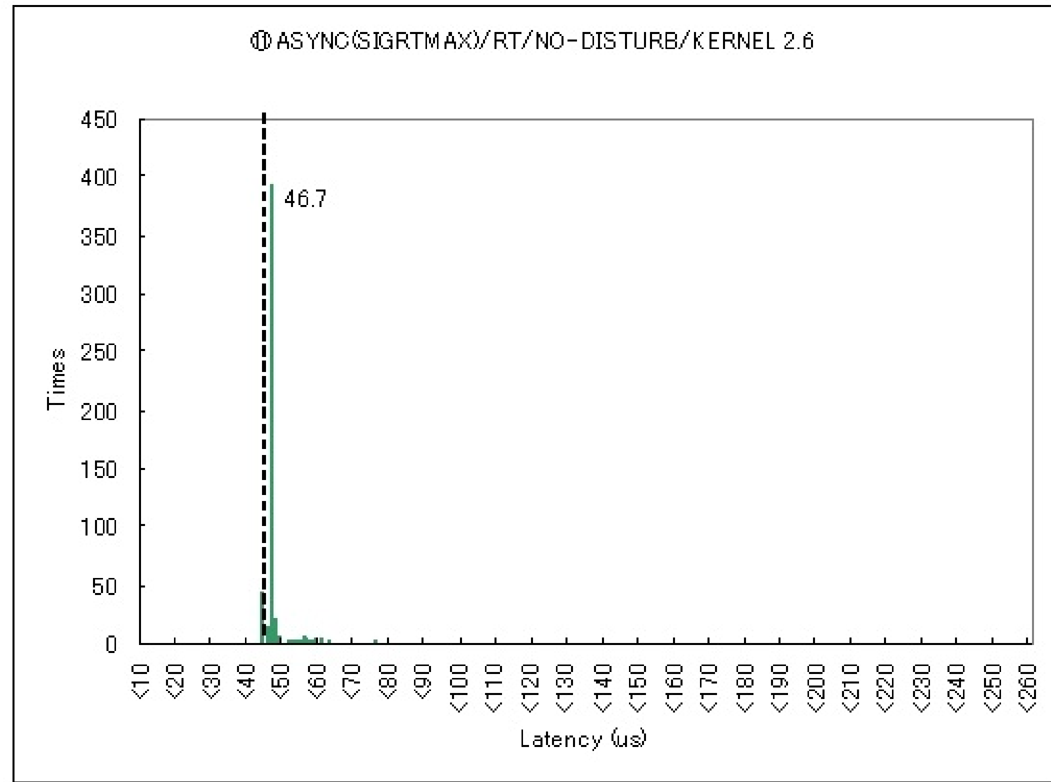

⑥ASYNC(SIGIO)/NO-RT/WITH DISTURB/KERNEL 2.6
2186.0
Times
Latency (us)
90
80
70
60
50
40
30
20
10
0
<10
<20
<30
<40
<50
<60
<70
<80
<90
<100
<110
<120
<130
<140
<150
<160
<170
<180
<190
<200
<210
<220
<230
<240
<250
<260

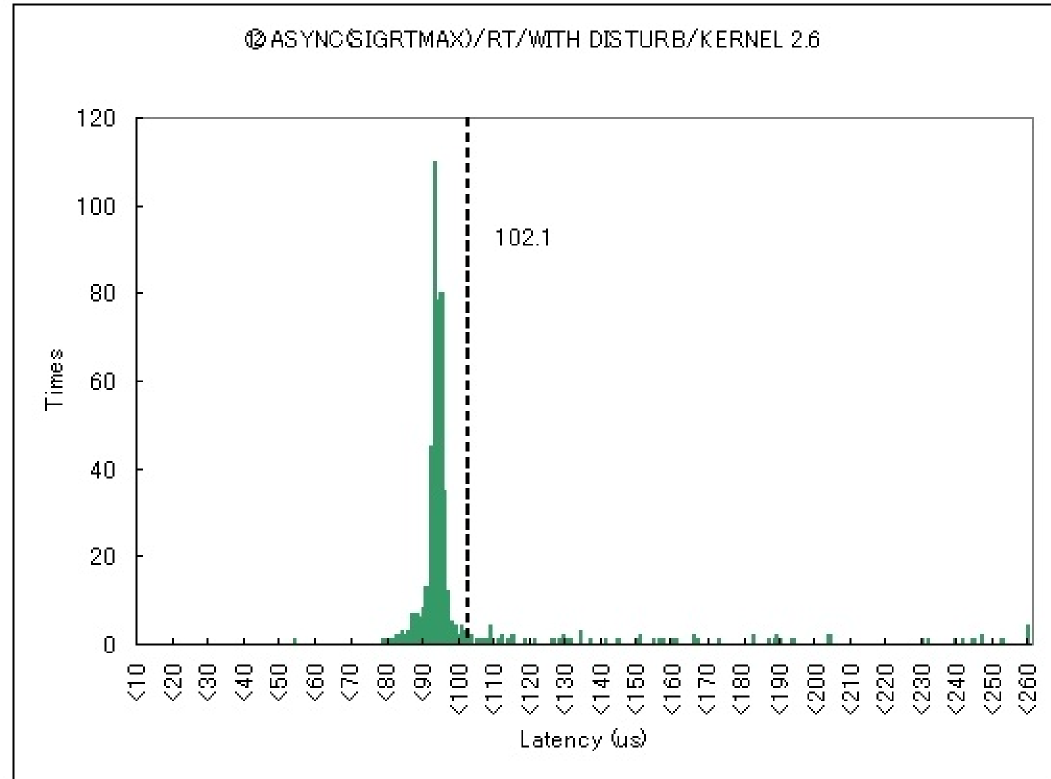

⑦ASYNC(SIGIO)/RT/NO-DISTURB/KERNEL 2.6
400
350
300
250
200
150
100
50
0
Times
47.1
<10
<20
<30
<40
<50
<60
<70
<80
<90
<100
<110
<120
<130
<140
<150
<160
<170
<180
<190
<200
<210
<220
<230
<240
<250
<260
Latency (us)

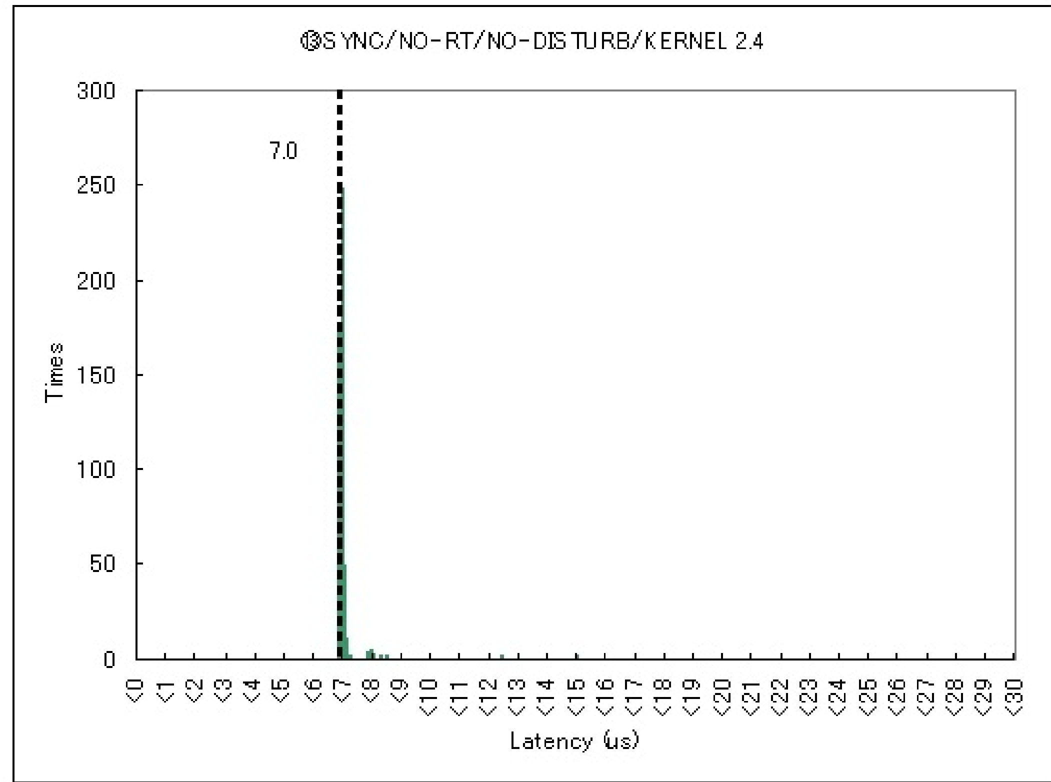

⑧ASYNC(SIGIO)/RT/WITH DISTURB/KERNEL 2.6
100
90
80
70
60
50
40
30
20
10
0
Times
102.9
<10
<20
<30
<40
<50
<60
<70
<80
<90
<100
<110
<120
<130
<140
<150
<160
<170
<180
<190
<200
<210
<220
<230
<240
<250
<260
Latency (us)

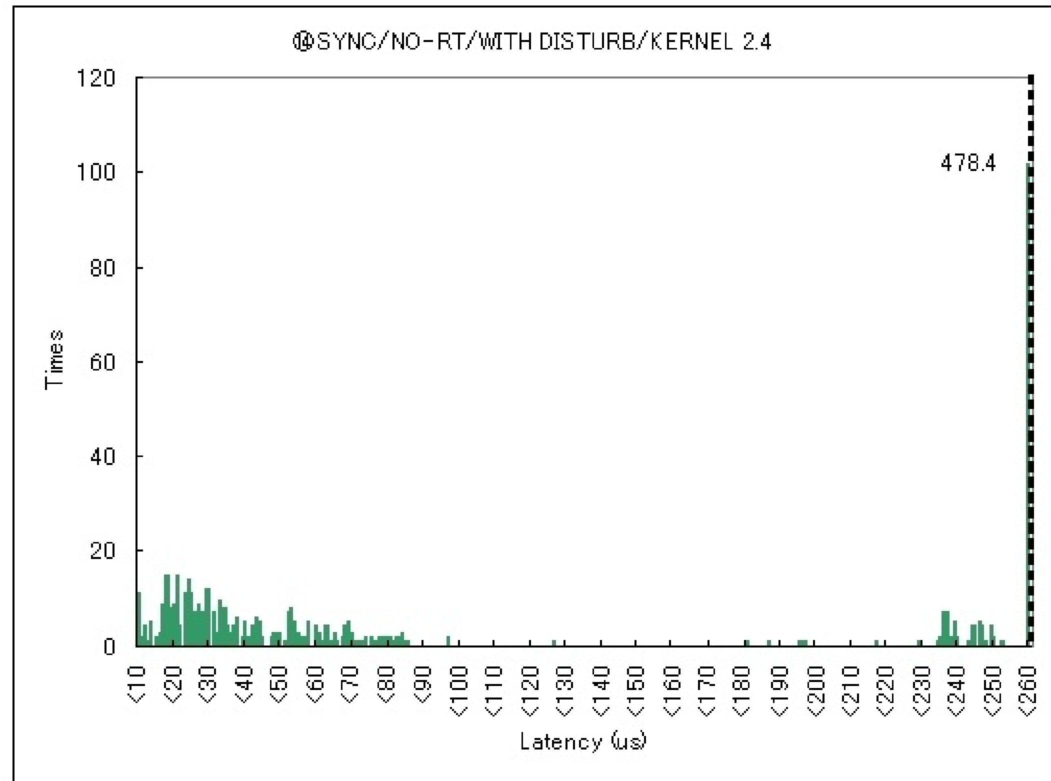

⑨ASYNC(SIGRTMAX)/NO-RT/NO-DISTURB/KERNEL 2.6
350
300
250
200
150
100
50
0
Times
50.0
<10
<20
<30
<40
<50
<60
<70
<80
<90
<100
<110
<120
<130
<140
<150
<160
<170
<180
<190
<200
<210
<220
<230
<240
<250
<260
Latency (us)

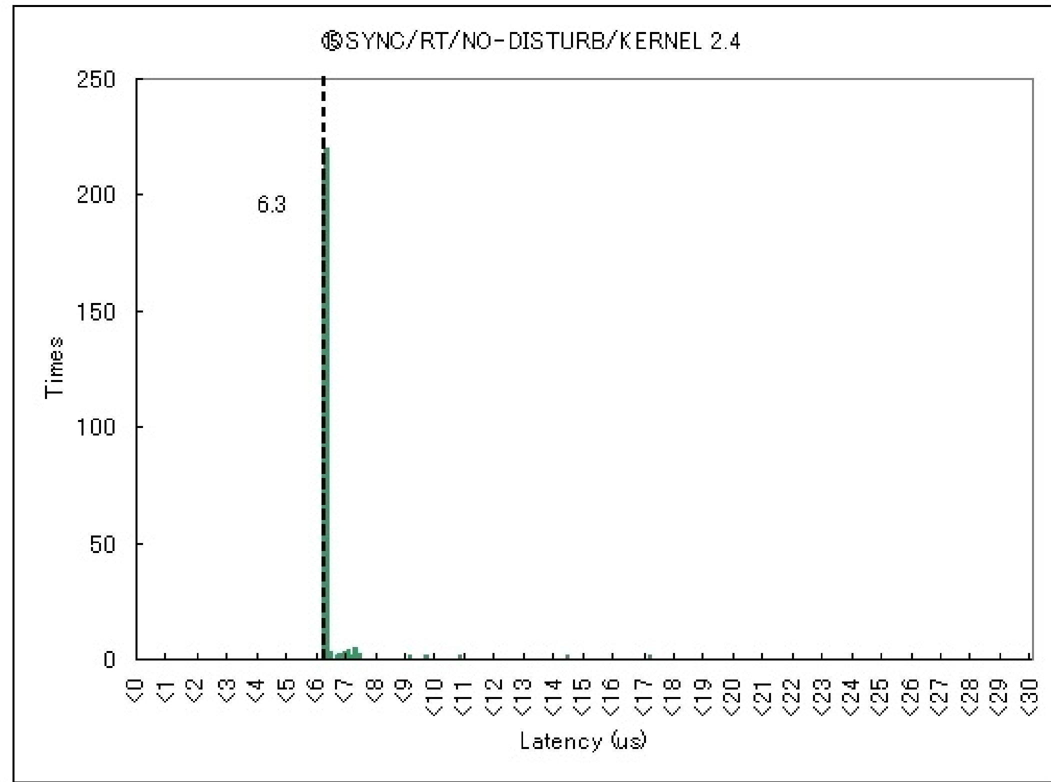

⑩ASYNC(SIGRTMAX)/NO-RT/WITH DISTURB/KERNEL 2.6
120
100
80
60
40
20
0
Times
2528.2
<10
<20
<30
<40
<50
<60
<70
<80
<90
<100
<110
<120
<130
<140
<150
<160
<170
<180
<190
<200
<210
<220
<230
<240
<250
<260
Latency (us)

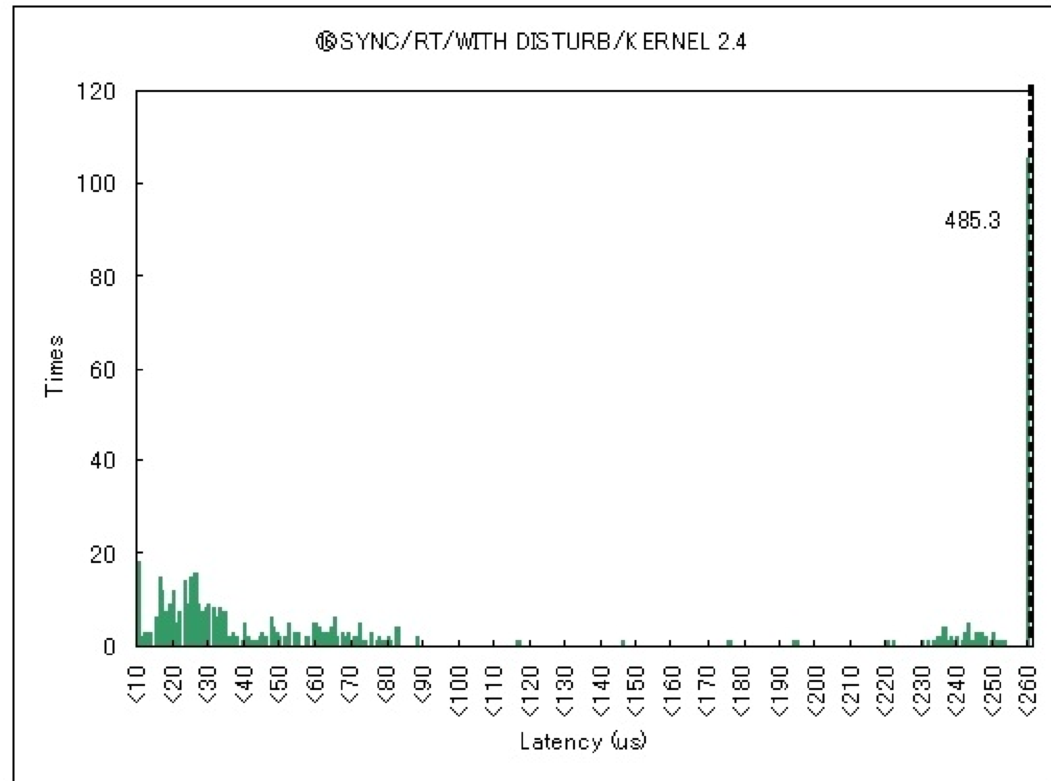

⑪ ASYNC(SIGRTMAX)/RT/NO-DISTURB/KERNEL 2.6
Times
Latency (us)
46.7
450
400
350
300
250
200
150
100
50
0
<10
<20
<30
<40
<50
<60
<70
<80
<90
<100
<110
<120
<130
<140
<150
<160
<170
<180
<190
<200
<210
<220
<230
<240
<250
<260


⑫ ASYNC(SIGRTMAX)/RT/WITH DISTURB/KERNEL 2.6
Times
Latency (us)
102.1
120
100
80
60
40
20
0
<10
<20
<30
<40
<50
<60
<70
<80
<90
<100
<110
<120
<130
<140
<150
<160
<170
<180
<190
<200
<210
<220
<230
<240
<250
<260


⑬SYNC/NO-RT/NO-DISTURB/KERNEL 2.4
7.0
Times
Latency (us)


⑭SYNC/NO-RT/WITH DISTURB/KERNEL 2.4
478.4
Times
Latency (us)


⑮SYNC/RT/NO-DISTURB/KERNEL 2.4
Times
250
200
150
100
50
0
6.3
<0 <1 <2 <3 <4 <5 <6 <7 <8 <9 <10 <11 <12 <13 <14 <15 <16 <17 <18 <19 <20 <21 <22 <23 <24 <25 <26 <27 <28 <29 <30
Latency (us)


⑯SYNC/RT/WITH DISTURB/KERNEL 2.4
Times
120
100
80
60
40
20
0
485.3
<10 <20 <30 <40 <50 <60 <70 <80 <90 <100 <110 <120 <130 <140 <150 <160 <170 <180 <190 <200 <210 <220 <230 <240 <250 <260
Latency (us)

> From: eLinux.org

SMC WSKP100

The SMC WSKP100 Wifi Skype phone is a Linux running phone meant to be used for Skype over wifi without having to u with a 1.8" CSTN LCD (128 x 160, 65K colors), Mini USB and earphone ports, various buttons/keys including volume and capable wifi module. You can buy the phone from Amazon.

The phone has a custom built Linux distribution made by SMC to run the Skype binaries. The goal of this project is to mal software besides Skype. Since the processor in the phone is ARM9 based, it could be used for some other more interesti SIP based phone or even a wifi mesh of wifi phones.

Known Components:

```
* TI OMAP1710
      o 220 MHz ARM926TEJ and MS320C55x DSP SoC
* TI TSC2101
      o 4-Wire Touch Screen Controller
      o St. DAC / Mono ADC with HP/Speaker Amplifier
* Samsung K4S56163PF
      o 4M x 16bit x 4 banks (256Mbit) SDRAM
* Intel 28F128L18B
      o 16MB NOR Flash
* TI TPS65013
      o 1-cell Li-Ion Power Mgmt IC
      o USB/AC Charger, 2DC/DC, 2 LDOs, I2C interface
      o http://www.ti.com/lit/gpn/tps65013
* Marvell 88W8385-BDK1
      o 802.11B/G WiFi
```

(source: http://wiki.gpl-devices.org/index.php/SMC_WSKP100G)

Status:

August 15, 2007: I'm currently awaiting for my phone to arrive. While I'm waiting, I'm gathering as much information about 1.8v serial port, I ordered samples of the MAX3218 so I can hook it up to a computer. Also, I was able to extract the binar provided by SMC by just running the tool in Windows and copying them out of c:\winnt\Temp\Skypyphone\Flasher before are encrypted. I also tried the same method with the firmware flasher from the very similar yet more expensive SMCWSP encrypted. This might prove useful as I could possibly re-use things from the SMCWSP-100 for this one and can most like the WSKP100.

(sources: http://spritesmods.com/?art=wskpsip and http://wiki.gpl-devices.org/index.php/SMC_WSKP100G)

August 16, 2007: I've succesfully mounted one of the jffs2 images from the firmware flasher of the SMCWSP-100 since th make two really simple bash scripts to simplify mounting and umounting jffs2 images since the process isn't so simple. Yo umount_jffs2. These scripts assume you're mounting/umounting one jffs2 image at a time. Obviously there is no guarante even work for you. You have been warned :-)

(sources: http://linux-7110.sourceforge.net/howtos/netbook_new/x1125.htm)

August 17, 2007: I've been going through the jffs2 images of the SMCWSP-100 and been finding some interesting things. the GUI. Another tidbit came from GPSFan in #edev (irc.freenode.net) that in /root/sbin/QCopSend there is references to leftovers from the original programmers/system designers have been found, which are always interesting to see. I'm hopi WSKP100, but I'll have to wait till I hook it upto a serial port and gain root access to the running system as its firmware im

Interesting Links and Hacks:

http://spritesmods.com/?art=wskpsip - A hack involving installing the firmware from SMCs Wifi SIP phone onto this phone

http://www.wifiphone24.com/download.aspx - SMC's Download page for firmware for the phone and GPL'd sources

http://www.smc.com/index.cfm?
event=downloads.doSearchCriteria&localeCode=EN_GBR&knowsPartNumber=false&productCategory=14&modelNumb
- Another Download page for firmware & source

> http://www.wifiphone24.com/tech_specification.aspx - SMC's Tech Specs for the phone
>
> http://linux.omap.com/ - Official TI OMAP Linux
>
> http://www.muru.com/linux/omap/ - Unofficial OMAP Linux
>
> http://free-electrons.com/articles/omap - More info on OMAP Linux
>
> http://linux.omap.com/pub/documentation/omap_1710v1.4.txt - Some interesting info on installing Linux on a OMAP1710
>
> http://www.nmacleod.com/nokia/schematics/N770_Schematics.pdf - Schematic of the Nokia 770 which shows pin descrip
>
> http://www-s.ti.com/sc/psheets/mpbg284a/mpbg284a.pdf - Pin Layout of the OMAP1710's BGA package
>
> http://git.infradead.org/?p=libertas-2.6.git;a=summary - Driver for the onboard Wifi module

---

--G1powermac [g1powermac@yahoo.com] 02:00, 18 August 2007 (EEST)

Category:

- Hardware Hacking

> From: eLinux.org

Sparkfun Camera

MagnaChip HV7131GP CMOS Camera Module

- 640x480 (actual 652 x 488)
- datasheet
- on sale at Sparkfun for $19.95
- SMD Connector is a DF18 Hirose Connector
- SMD Connect available from Sparkfun for $0.95
- Break out Board
- SparkfunCameraFPGA
- AVR project based on a similiar cameras - http://www.jrobot.net/Projects/AVRcam.html

8-Bit Output Format Support:

- `YCbCr 4:2:2`
- `Bayer`

the libdc1394 howto provides an excellent introduction to the Bayer format.

the starting row and column determine the Bayer format

- even `BGGR`
- odd `GRBG`

basically because the camera is slightly larger than 640x480 you can set the window for usage:

- Row 2 and Column 2 with Window Heigh 640 and Width 480 produces `Bayer BGGR`
- Row 1 and Column 1 with Window Heigh 640 and Width 480 produces `Bayer GRBG`

the odd/even are dependent on the orientation of the scan.

Information gathered from the Sparkfun Forums

although the camera does support a 16-bit mode, the ribbon cable only provides support for the 8-bit mode.

```
          ------------
     ENB  | 11      1 |    Y0
     RST  | 12      2 |    Y1
     GND  | 13      3 |    Y2
     VDD  | 14      4 |    Y3
     SCL  | 15      5 |    Y4
     SDA  | 16      6 |    Y5
  FVALID  | 17      7 |    Y6
  LVALID  | 18      8 |    Y7
    PCLK  | 19      9 |    VDD
    MCLK  | 20     10 |    GND
          ------------

NOTE: Pins 14 + 9 are connected to VDD :
V+ --|
     |
     |----[B1]--------------#--------- Pin 9
     |                      |
     |                      |
     |----[B2]-------#------)--------- Pin 14
                     |      |
                     |      |
                    [C1]   [C2]
                     |      |
                   -----  -----
                    ---    ---
                     -      -

busonerd states - "My guess for the parts values would be a ~600ohm @ 100MHZ ferrite bead (B1 and B2), current capaci
ty ~100-200 ma, and a 0.1uf caps(C1 and C2). You really can't go wrong with the ferrite bead - just get a higher impe
dance + higher current capacity one if in doubt. For the caps(C1 and C2) - if you really want to be paranoid, use an
0.47uf, an 0.1uf and an 0.01uf - with the smaller values nearer the connector - but that is guaranteed overkill."
```

- the i2c interface needs to be level shifted from the 2.8v to either 3.3v or 5v depending on your host device
- http://www.standardics.philips.com/support/documents/i2c/pdf/an97055.pdf
  - application note on i2c level shifting

Category:

- Hardware

> From: eLinux.org

TCube Info

This page describes how to load Linux on the T-Cube board.

Table Of Contents:

Contents

Hardware Used

- Board Order Number: SEMC5701(T-Cube)/SHIMAFUJI
  - CPU
    - VR5701 (NEC VR5500A SOC chip) (333MHz)
- Memory
  - 64 Mbyte SDRAM
  - 16 Mbyte Flash
- Graphics controller
  - SMI (SM722GX)
- Other Feature
  - Timer x 4 (VR5701)
  - UART x2 (VR5701 /16550 compatible)
  - PCI (VR5701) -- no PCI connector/header on the baord
  - Sound AC97 (VR5701)
  - IDE/Compact Flash (VR5701)
  - RTC (RV 5C348B)
  - Ethernet (Intel GD82559ERSL3DG)
  - USB(USB2) (NEC uPD720101)
  - CF

Software and Configuration Files Used

- PMON
  - setup
  - boot

- Toolchain
  - Mips Toolchain from Lineo
- kernel
  - celinux-040503.tar.bz2
- root filesystem
  - [unprepared]

General Outline for Bootstrapping using PMON

Preparing Target Hardware and Connections

-
  - Connections:
    - Host machine Ethernet to hub (I configured as 192.168.1.1)
    - T-Cube Ethernet to hub (I configured as 192.168.1.2)
    - Connect serial cable between T-Cube and host machine
      - Serial1 connector to interface module serial port
      - Other end of null-modem cable to first serial port (ttyS0) on the Linux host machine

Preparing Host Software

- Download and place files in /tftpboot
- Make sure tftpd is ready to run
- Confirm minicom is configured to access /dev/ttyS0, at 115K N81

Using PMON to Boot the Kernel

- Root filesystem is NFS

Building the Kernel

Here are some commands I used to build the kernel:

- cd celinux-040503/
- export PATH=/usr/local/bin:$PATH
- cp arch/mips/defconfig-tcube .config
- make ARCH=mips CROSS_COMPILE=mipsel-linux- menuconfig
  - disable XIP
- make ARCH=mips CROSS_COMPILE=mipsel-linux- dep
- make ARCH=mips CROSS_COMPILE=mipsel-linux-

Preparing the Kernel to Download

- cp vmlinux /tftpboot/

Starting the NFS daemon

- /etc/rc.d/init.d/nfs start

Setting Environment Variable and Write Environment Value on T-Cube

- minicom (set to /dev/ttyS0)
- Set environment variable
  - Set variable tty0 (I used "set tty0 115200")

    - Set variable netaddr (I used "set netaddr 192.168.1.2")
    - Set variable bootaddr (I used "set bootaddr 192.168.1.1")
    - Set variable bootfile (I used "set bootfile vmlinux")
- Write environment value
    - Employed "write_env" command

Downloading the Kernel to T-Cube

- minicom (set to /dev/ttyS0)
- Download the kernel
    - Employ "boot" command
    - As "boot" command completes, "Entry-address" appears.

Booting the Kernel

- Employ "go" command (I used "g -e Entry-address")

Category:

- Development Boards

From: eLinux.org

TUSB2046B

TUSB2046B is a RoHS Compliant single chip 4-Port USB Hub from Texas Instruments

```
* Fully Compliant With the USB Specificationas a Full-Speed Hub: TID #30220231
* 32-Terminal LQFP Package With a 0.8-mm Terminal Pitch or QFN Package with a 0.5-mm Terminal Pitch(1)
* 3.3-V Low Power ASIC Logic
* Integrated USB Transceivers
* State Machine Implementation Requires No Firmware Programming
* One Upstream Port and Four Downstream Ports
* All Downstream Ports Support Full-Speed and Low-Speed Operations
* Two Power Source Modes
   * Self-Powered Mode
   * Bus-Powered Mode
* Power Switching and Overcurrent Reporting Is Provided Ganged or Per Port
* Supports Suspend and Resume Operations
* Supports Programmable Vendor ID and Product ID With External Serial EEPROM
* 3-State EEPROM Interface Allows EEPROM Sharing
* Push-Pull Outputs for PWRON Eliminate the Need for External Pullup Resistors
* Noise Filtering on OVRCUR Provides Immunity to Voltage Spikes
* Package Pinout Allows 2-Layer PCB
* Low EMI Emission Achieved by a 6-MHz Crystal Input
* Migrated From Proven TUSB2040 Hub
* Lower Cost Than the TUSB2040 Hub
* Enhanced System ESD Performance
* Supports 6-MHz Operation Through a Crystal Input or a 48-MHz Input Clock
```

TUSB2046B Product Page

Mouser Part Lookup

Digikey Part Lookup

Jameco Part Lookup

Custom Hub Project using a TUSB2046

Category:

- Hardware

> From: eLinux.org

TvNow



U1 Sharp LH79524

```
    ARM720T Processor
```

U2 Conexant CX25836-3A

```
    Located below a shield. TQFP-64
    Single chip video decoder. http://elinux.org/images//8/83/Pdf.gif Datasheet http://elinux.org/images/d/da/Info_cir
cle.png and http://elinux.org/images//8/83/Pdf.gif Product Brief http://elinux.org/images/d/da/Info_circle.png.
    IO are LVTTL. IOL=4mA drive.
```

U4 Unknown Chip On Board(CoB) package U5 Green unmarked 0805 package

```
    Located on USB board
```

U6 Green unmarked 0805 package

```
    Located on USB board
```

U7 Unknown Chip On Board(CoB) package U8 TI LM324 U9 Hynix HY57V643220DT-6

```
    Long, leaded TSOP II 86 pin package
    SDRAM, VDD=3.3 & VDDQ=3.3V, 64Mbit, x32, 4banks, LVTTL
    Decoded per the http://elinux.org/images//8/83/Pdf.gif datasheet http://elinux.org/images/d/da/Info_circle.png. 6n
s (166MHz), 2Mx32
```

U11 Hynix HY27UA081G1M

```
    Wide, leaded package
    NAND flash, 1Gb
```

U12 U14 Unknown Chip On Board(CoB) package

Boards The entire system is composed of 4 boards. Most electrolytics are through hole. However, there are a few scattered tantalums.

```
    * There is a main board with the CPU, memory, and flash, and a SD socket. There is a cylinder shaped crystal (prob
ally 32KHz, watch xtal) and a 28.6363Mhz xtal.
    * There is a board that has the AV inputs on one side.
    * On the otherside, there is a board with a mini USB connector.
    * A 4th board sits on top of 2 sets of fine (0.05 or finer) headers.
```

discusion: http://forums.parallax.com/forums/default.aspx?f=15&m=158549

vendor: http://www.hasbro.com/tiger/default.cfm?page=browse&product_id=18617



Category:

- Hardware Hacking

> From: eLinux.org

VGF-CP1

Contents

Sony VAIO Canvas Online - Digital Picture Frame

http://bb.watch.impress.co.jp/cda/static/image/2008/05/07/cp101s.jpg

specification in brief:

- Display Size: 7″
- Pixel Format: 800 x 480
- Internal Memory: 100MB (80MB for pictures, 15MB for audio)
- External Storage: CF, MS, SD
- Image Support: JPEG, BMP, PNG, GIF
- Audio Support: MP3, WAV
- CPU: 400Mhz ARM11 (i.MX31)
- OS: Timesys Linux 2.6.19
- Wireless: 802.11b/g
- Availability: May 17, 2008

Firmware

Though the CP1 was available in the US and UK, there seems to be no english firmware for download available. Two japanese firmware images are available. You can flash them by just placing the .IMG file in a sub-folder called 'Update'.

```
VGF-CP1 japanese firmware Ver.2.00.00.09032516: http://vcl.vaio.sony.co.jp/download/EP0000177017.html
.IMG File: http://dlv.update.sony.net/pub/vaio/download/EP0000177017/EP0000177017.img

VGF-CP1 japanese firmware Ver.1.01.00.08082520: http://vcl.vaio.sony.co.jp/download/EP0000162984.html
.IMG File: http://dlv.update.sony.net/pub/vaio/download/EP0000162984/EP0000162984.img
```

Bootlog

Captured with serial interface (115200,8,n,1) on MX31 UART1.

``

```
 Linux 2.6 Freescale MXC processor


 Choose an option from below:
```

```
1. Load kernel to RAM and then boot from [0x80008000]
2. Change the Linux kernel destination loading address [0x80008000]
3. Enter command line option for kernel
4. Change command line option address [0x80000100]

  Please enter selection ->   Timeout occured



-->Booting from RAM...
RAM size is 128MB

Copying pages....

-->Goint Start

Copying pages....

-->Goint Start
-->Starting kernel...
Uncompressing Linux.........................................................................................
........... done, booting the kernel.
Linux version 2.6.19.2 (akatsuka@akatsuka2.sm.sony.co.jp) () #1 PREEMPT Fri Apr 25 10:29:51 JST 2008
CPU: Some Random V6 Processor [4107b364] revision 4 (ARMv6TEJ), cr=00e5387f
Machine: Sony IDAM
Memory policy: ECC disabled, Data cache writeback
CPU0: D VIPT write-back cache
CPU0: I cache: 16384 bytes, associativity 4, 32 byte lines, 128 sets
CPU0: D cache: 16384 bytes, associativity 4, 32 byte lines, 128 sets
Built 1 zonelists.  Total pages: 32128
Kernel command line: initrd=0x83000000,8M root=/dev/ram0 rw console=ttymxc0,115200 ip=on
MXC IRQ initialized
PID hash table entries: 512 (order: 9, 2048 bytes)
Actual CLOCK_TICK_RATE is 16625000 Hz
Console: colour dummy device 80x30
Dentry cache hash table entries: 16384 (order: 4, 65536 bytes)
Inode-cache hash table entries: 8192 (order: 3, 32768 bytes)
Memory: 64MB 64MB = 128MB total
Memory: 118072KB available (2416K code, 601K data, 472K init)
Mount-cache hash table entries: 512
CPU: Testing write buffer coherency: ok
checking if image is initramfs...it isn't (no cpio magic); looks like an initrd
Freeing initrd memory: 8192K
NET: Registered protocol family 16
MXC GPIO hardware
system_rev is: 0x20
Clock input source is 26000000
Machine Type: iDAM
L2 cache: WB
Using SDMA I.API
MXC DMA API initialized
usb: Host 2 registered
SCSI subsystem initialized
CSPI: mxc_spi-0 probed
CSPI: mxc_spi-1 probed
usbcore: registered new interface driver usbfs
usbcore: registered new interface driver hub
usbcore: registered new device driver usb
MXC I2C driver
NET: Registered protocol family 23
NET: Registered protocol family 2
IP route cache hash table entries: 1024 (order: 0, 4096 bytes)
TCP established hash table entries: 4096 (order: 2, 16384 bytes)
TCP bind hash table entries: 2048 (order: 1, 8192 bytes)
TCP: Hash tables configured (established 4096 bind 2048)
TCP reno registered
Low-Level PM Driver module loaded
JFFS2 version 2.2. (NAND) (SUMMARY)  (C) 2001-2006 Red Hat, Inc.
io scheduler noop registered
io scheduler anticipatory registered
io scheduler deadline registered
io scheduler cfq registered (default)
```

```
Console: switching to colour frame buffer device 100x30
mxcfb: fb registered, using mode SAMSUNG-WVGA-32
MXC WatchDog Driver 2.0
MXC Watchdog Timer: initial timeout 60 sec
Serial: MXC Internal UART driver
mxcintuart.0: ttymxc0 at MMIO 0x43f90000 (irq = 45) is a Freescale MXC
mxcintuart.2: ttymxc2 at MMIO 0x5000c000 (irq = 18) is a Freescale MXC
RAMDISK driver initialized: 16 RAM disks of 16384K size 1024 blocksize
loop: loaded (max 8 devices)
MXC MTD nor Driver 2.0
mxc_nor_flash: probe of mxc_nor_flash.0 failed with error -5
MXC MTD nand Driver 2.0
NAND device: Manufacturer ID: 0xad, Chip ID: 0xdc (Hynix NAND 512MiB 3,3V 8-bit)
Scanning device for bad blocks
Bad eraseblock 732 at 0x05b80000
Bad eraseblock 1450 at 0x0b540000
Bad eraseblock 2678 at 0x14ec0000
Bad eraseblock 2680 at 0x14f00000
Bad eraseblock 3981 at 0x1f1a0000
Creating 6 MTD partitions on "NAND 512MiB 3,3V 8-bit":
0x00000000-0x00040000 : "IPL-SPL"
0x00040000-0x00440000 : "nand.kernel"
0x00440000-0x00c40000 : "nand.rootfs"
0x00c40000-0x02c40000 : "nand.userreg"
0x02c40000-0x06c40000 : "nand.userprog"
0x06c40000-0x20000000 : "nand.userdata"
fsl-ehci fsl-ehci.0: Freescale On-Chip EHCI Host Controller
fsl-ehci fsl-ehci.0: new USB bus registered, assigned bus number 1
write priority to fc004400 302154
iram0 = 1ffc0000 iram1 = 1ffc0800
fsl-ehci fsl-ehci.0: irq 36, io base 0x43f88400
fsl-ehci fsl-ehci.0: USB 2.0 started, EHCI 1.00, driver 10 Dec 2004
usb usb1: configuration #1 chosen from 1 choice
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 1 port detected
Initializing USB Mass Storage driver...
usb 1-1: new high speed USB device using fsl-ehci and address 2
usb 1-1: configuration #1 chosen from 1 choice
hub 1-1:1.0: USB hub found
hub 1-1:1.0: 2 ports detected
usb 1-1.1: new high speed USB device using fsl-ehci and address 3
usb 1-1.1: configuration #1 chosen from 1 choice
scsi0 : SCSI emulation for USB Mass Storage devices
usbcore: registered new interface driver usb-storage
USB Mass Storage support registered.
usbcore: registered new interface driver asix
MXC keypad loaded
input: mxckpd as /class/input/input0
IPU Post-filter loading
SSI module loaded successfully
Advanced Linux Sound Architecture Driver Version 1.0.13 (Tue Nov 28 14:07:24 2006 UTC).
Control ALSA component registered
MXC audio support initialized
ALSA device list:
  #0: MXC audio for iDAM
TCP cubic registered
NET: Registered protocol family 1
NET: Registered protocol family 17
VFP support v0.3: implementor 41 architecture 1 part 20 variant b rev 2
RAMDISK: Compressed image found at block 0
VFS: Mounted root (ext2 filesystem).
Freeing init memory: 472K
Using fallback suid method
Using fallback suid method
Using fallback suid method
Using fallback suid method
#/etc/rc.d/rcS start
Using fallback suid method
usb_address[0] 3
scsi 0:0:0:0: Direct-Access     Sony     MCR-U HS-MS      5.58 PQ: 0 ANSI: 0
sd 0:0:0:0: Attached scsi removable disk sda
scsi 0:0:0:1: Direct-Access     Sony     MCR-U HS-CF      5.58 PQ: 0 ANSI: 0
```

```
sd 0:0:0:1: Attached scsi removable disk sdb
scsi 0:0:0:2: Direct-Access     Sony     MCR-U HS-SD      5.58 PQ: 0 ANSI: 0
k3d_chk_one: ejected, device sda
k3d_chk_one: ejected, device sdb
sd 0:0:0:2: Attached scsi removable disk sdc
k3d_chk_one: ejected, device sdc
ysdev: module license 'unspecified' taints kernel.
ysdev: Yosemite misc driver (Mar 13 2008)
GPIO port 2 (0-based), pin 31 is already reserved!
GPIO port 2 (0-based), pin 31 is already reserved!
EnablePA=0x01
PA Group #0: level=13dbm, Rate Bitmap=0x1800 (54 48 Mbps)
PA Group #1: level=14dbm, Rate Bitmap=0x7e0 (36 24 18 12 9 6 Mbps)
PA Group #2: level=16dbm, Rate Bitmap=0x0f (11 5.5 2 1 Mbps)
eth0      version:sd2480-9.70.3.p20-26409.p43 (r208)
#/usr/sony/etc/rcS start
CLCDC_Init: major device 251
CAMERA_Init: major device 250
Loaded PowerVR consumer services.
command : time

Get time val : 1199232528
set system clock : date 010200082008.48(1199232528)
Wed Jan  2 00:08:48 UTC 2008
Yostart  was started in dll [2]
Yostar:writeing pid=1064
Yostart with CHECK was started in dll [2]
Yostart  was started in dll [6]
Yostar:writeing pid=1066
Yostart with CHECK was started in dll [6]
ysem.c: 230: ysem_open: k != -1 failed
ysem.c: 230: ysem_open: k != -1 failed
Yostart  was started in dll [42]
Yostar:writeing pid=1069
Yostart with CHECK was started in dll [42]
zero clear memory
ysem.c: 230: ysem_open: k != -1 failed
ysem.c: 230: ysem_open: k != -1 failed
ADD 0 MTX: msgc_alive_mtx_00, SEM: msgc_result_sem_00
Yostart  was started in dll [45]
Yostar:writeing pid=1073
Yostart with CHECK was started in dll [45]
Yostart with CHECK was started in dll [30]
Yostart  was started in dll [30]
Yostar:writeing pid=1076
0:08:48 PID1076 event.c( 401)event_open_mem       : failed open_share()
Yostart  was started in dll [5]
Yostar:writeing pid=1079
Yostart with CHECK was started in dll [5]
ysem.c: 230: ysem_open: k != -1 failed
ADD 1 MTX: msgc_alive_mtx_01, SEM: msgc_result_sem_01
Yostart  was started in dll [48]
Yostar:writeing pid=1083
Yostart with CHECK was started in dll [48]
ysem.c: 230: ysem_open: k != -1 failed
ADD 2 MTX: msgc_alive_mtx_02, SEM: msgc_result_sem_02
[vunp] 1 hour sleep!!
Yostart  was started in dll [49]
Yostar:writeing pid=1091
Yostart with CHECK was started in dll [49]
ysem.c: 230: ysem_open: k != -1 failed
ADD 3 MTX: msgc_alive_mtx_03, SEM: msgc_result_sem_03
[nled] write : ledset -s -b 80 0

[systemapi] local_time ti:0  time_st:0  time_js:3600  time_ut:0
ontimer set event!!!!
ontimer set event!!!!
ontimer set event!!!!
ontimer set event!!!!
Yostart with CHECK was started in dll [51]
Yostart  was started in dll [51]
Yostar:writeing pid=1095
```

```
ysem.c: 230: ysem_open: k != -1 failed
ADD 4 MTX: msgc_alive_mtx_04, SEM: msgc_result_sem_04
Yostart with CHECK was started in dll [52]
Yostart  was started in dll [52]
Yostar:writeing pid=1099
Yostart with CHECK was started in dll [10]
Yostart  was started in dll [10]
Yostar:writeing pid=1103


[terahttp] dll_init
MAIN_LOOP_START_REQ
MAIN_LOOP_START_END
Yostart  was started in dll [12]
Yostar:writeing pid=1106
Yostart with CHECK was started in dll [12]
Yostart with CHECK was started in dll [23]
Yostart  was started in dll [23]
Yostar:writeing pid=1108
pmb_start
pmb_loop
Message send Success. reply=1
region_code=0x41
mode=0x60
WPA2-PSK CCMP
error:[commontool.c]:L1373 -Mar 25 2009 ret=-5
ssid=<yournetworkname>
pairwise=CCMP
key=<yournetworkkey>
udhcpc (v0.9.9-pre+autoip) started
udhcpc[1230]: udhcpc (v0.9.9-pre+autoip) started
Sending discover...
udhcpc[1230]: Sending discover...
Sending select for 192.168.x.x...
udhcpc[1230]: Sending select for 192.168.x.x...
Lease of 192.168.x.x obtained, lease time 86400
udhcpc[1230]: Lease of 192.168.x.x obtained, lease time 86400
deleting routers
route: SIOC[ADD|DEL]RT: No such process
adding dns 192.168.y.y
!!!enter ntp callback
##### ntp: get network change
ntp set on
Message send Success. reply=1
0:09:00 PID1079 event.c( 969)event_api_add        : -------------------------------
0:09:00 PID1079 event.c( 970)event_api_add        : TERAAPI_REQ_ADD  registerd ID:3 data:[0]
0:09:00 PID1079 event.c( 971)event_api_add        : 2008/01/03(THU)10:28:26
0:09:00 PID1079 event.c( 972)event_api_add        : -------------------------------

Freescale Semiconductor, Inc.
i.MX31 Applications Development System

(none) login: terahttpd bind successfully

Freescale Semiconductor, Inc.
i.MX31 Applications Development System

(none) login:
```

Remote Access

Since firmware Ver.2.00 a web server is set up - however in japanese. Initial login is

```
ユーザー名 (loginname): root
パスワード (password): password
```

The web server allows you to configure your account data more easily than with the remote control. Unfortunately this login and password is not working on the serial console. With a root access account one might be able to add english translations.

No other services seems to run as all ports are closed.

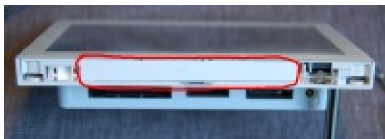
Hardware Disassembly

WARNING ! Disassembling the VGF-CP1 may cause damage ! It's on your own risk !!



remove two screws to seperate acrylic glass

1. To open the nice housing, you first have to remove two white square shaped sticker on the lower back side. The screws fix two white bushings, which have to be taken out in order to seperate the acrylic glass base.



remove sticker to open cover

1. Next you have to remove a rectangular white sticker. which is used as a diffusor for the blue led.



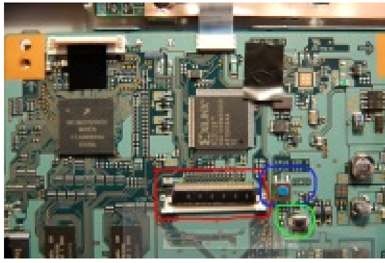
slightly shift display in arrow direction

1. To seperate the cover, you have to move carefully the display towards the direction marked with red arrows. The ribbon cable is on the bottom side, so you can open the housing now.



next remove 6 screws to open shield

1. The PCBs are shielded with a blank sheet fixed with six screws. For those you need well fitting screw driver, because the four ones marked in red are tightend very hard. I had to drill them out !



mainboard with display connector

1. Front display is connected to ribbon connector (red square). Another pwr button is inside (blue marking), next to it a reset button (green marking).



UART1 serial connector

1. A serial console can be connected through UART1 CN853. I already soldered another plug on top of the original one. Pinout see below. You'll need a level shifter LVTTL-to-RS232 as well in order to connect a PC serial port.
2. LVTTL/RS232 - Building a Custom Serial Interface http://buffalo.nas-central.org/wiki/Building_a_Custom_Serial_Interface

Pin 1 of the connector the left one on the picture and it is marked on the PCB with a white dot.

MX31 UART1 CN853 Pinout

Pin	Signal
1	Power (3.3V)*
2	Transmit (TX)
3	Receive (RX)
4	Ground (GND)

External Links

- Unboxing video: http://www.youtube.com/watch?v=oatks-1x8U8
- GPL software from Sony: http://www.sony.net/Products/Linux/VAIO/VGF-CP1.html
- Discussion thread: http://www.artiss.co.uk/2009/09/sony-vaio-vgf-cp1-photo-frame

Categories:

- Hardware
- Firmware
- Boot Time
- Hardware Hacking
- HOWTOs

From: eLinux.org

Wavefinder

Contents

DABUSB Wavefinder Information - Version 0.5

Wavefinder is a software-defined DAB radio designed jointly between Psion, RadioScape, Roke, Lost Wax and Real Networks; and marketed by Psion Infomedia circa year 2000. Wavefinder was originally a Windows-only product - the eventual aim of this project is to develop Free/Open Source GNU/Linux software, drivers and firmware suitable for controlling the Wavefinder from an embedded Linux platform such as the BeagleBoard.

In the future it might even be possible to replace the Wavefinder hardware with Open Hardware that takes advantage of modern highly integrated ICs (such as the TEMIC U2731B / Atmel ATR2731 or the Hitachi HD155080 tuner ICs) and using the DSP onboard the OMAP IC rather than 2x5402s.

Todo

- Remove any rubbish that's been superseded (ie. stuff we've figured out since).

Hardware

Main section (under shielding on USB cable side)

- 2x TMS320VC5402 - Texas Instruments 16 bit 5402 DSPs, DVC 5402GGU CD-0AA19DW
- ScanLogic SL11R-100
  - USB controller / 16 bit RISC processor (User Manual)
- 2x K6R1016V1C
  - 64K x 16 Bit High-Speed CMOS Static RAM(3.3V Operating)
- ADC1173 EM06AB - 8 bit 3V 15MSPS 33mW A/D converter
- MAX5541 - 16 bit serial DAC
- LMC6462BIM - Dual Rail to Rail Op-Amp
- TL431 - Programmable Precision Reference
- 2x LCX74PDXP - Dual D-type Flipflop
- HCT244
  - 8 bit buffer
- XILINX XCR3128A - Coolrunner PLD (programmable logic device)
- IDT QS3VH245Q XQ0041D - 8-bit bus switch (link)
- NC7S04 - Single CMOS inverter (marked 7S04)
- 48.000 MHz Crystal Oscillator / sytem clock
- 16.384 MHz device, marked 0047SM1. Reference clock?

The two TI 5402 DSPs provide about 200 MIPS of grunt. According to press releases from TI (link, link), these perform a limited amount of demodulation/decoding of the raw RF signal. More recent designs as used in the ModularTech PCI card, perform much more of the decoding on-board using a TMS320DRE200 (or TMS320C5416) DSP, and less on the host PC.

Radio section (large shield on reverse side)

- LMX2331 - Freq synthesiser or PLL (2 GHz & 510 MHz)
- LMX1511 - Freq synthesiser or PLL (1.1 GHz)
- Hitachi HWSD231
  - 38.912 MHz SAW filter (product page, product page)

Other bit (small shield on reverse side)

- 24LC16B
  - 2Kx8 Serial EEPROM (prob interfaced to USB controller)
- MAX1692 - Low-Noise, 5.5V Input, PWM Step-Down Regulator (marked 1692 EUB)
- Si9424DY - P-channel 20V (D-S) mosfet (marked 9424 JZA Y02C)

Power Regulation and Management (Revised 22/11/02)

There is a plethora of power supply devices. It looks like a MAX1692 PWM regulator (rated 600mA) provides a 3.3V supply from the USB 5V supply. This almost certainly provides the power for the SL11R and probably the DSPs and Xilinx chip as well. It also feeds an adjustable linear regulator set to 1.8V (LM1117MPX-ADJ - SOT223 package marked 'NO3A') which probably forms the core supply for the TMS320C5402's. There is also a mosfet (the Si9424) which appears to switch the unregulated power from the mains unit to a ZSR800 8V linear regulator which supplies a ZSR500 5V linear regulator. This regulator chain looks like it supplies the RF section (and, of course, the (in)famous LEDs). There is also a TPS76338 3.3V linear regulator (the SOT23-5 package marked 'PBEI'). This is fed directly from the Si9424 mosfet and here we see a (the?) reason for the Wavefinder's poor reliability. The TPS76338 has a maximum input voltage of 10V according to the data sheet and that is what the loaded unregulated supply provides - this means that this regulator is operating at the edge of its

specifications and is liable to fail - or at least it had on mine! The reason whoever designed it used a 10V PSU is that the ZSR800 has a minimum input voltage requirement of 10V. Why the TPS76338 isn't fed from either the ZSR800 or the ZSR500 I don't know - they're both rated at 200mA.

SL11R microcontroller

The SL11R's memory map:

```
Int RAM       0000-0BFF (3k)
Ext RAM       0C00-7FFF (29k)
Ext Pg.1/DRAM 8000-9FFF (8k)
Ext Pg.2/DRAM A000-BFFF (8k)
Mem map reg   C000-C0FF (256b)
Ext ROM       C100-E7FF (9984b)
Int ROM       E800-FFFF (6k)
```

It seems that the memory mapped registers (0xC000-0xC0FF) are mapped directly from USB commands such as "e6 c0 00 00", etc. The processor is 16-bit, and data words are sent LSB first. This holds true for commands with the first word being in the range (0xC000-0xC0FF). It *may* be that some of the other commands are writing direct to memory.

General software tools

Psion's Wavefinder software

The basis for reverse-engineering the Wavefinder, the Psion software provides the libraries used by any of the Wavefinder control software (e.g. native Psion software, DABBar, etc.). It also provides an API for use by custom programs such as Dabble (extrememly useful when combined with hooking the DLL functions).

Files

- **rsDSP[*ab*].bin** - contain the DSP code. The data from both of these files is sent to the Wavefinder at startup.
- **rsCPU***\.dll** - different version of CPU-intensive code for different processors
- **ViadabReceiver.dll** - lots of stuff, including LED control code

Snoopy/SnoopyPro

This is a USB sniffer, logging all tranfers over the USB bus related to particular devices.

Custom software tools

Dabble

Dabble uses the Radioscape API to control the Wavefinder while hooking some DLLs. This allows the behaviour of the supplied Wavefinder software and what it sends over the USB bus to be studied. (maybe something on interpreting dabble output)

PAD extraction

Kristoff Bonne has written a PAD extractor. pad_decode and tdc_decode. (url) (add more details)

wlights

Controls the Wavefinder's LEDs independently of the Radioscape DLLs.

wboot

Further development of the wlights code so that now it will bootup the wavefinder, download the DSP code, tune it to 'something', and collect all the data returned. No real attempt so far to understand the content of the data streams, other than the semi-obvious.

find_wavefinder() - Now uses some setupapi calls to find the wavefinder driver, based on its GUID. My PC seems to like both {96cb3fae-594e-11d3-b317-00e02914a689} and {a5dcbf10-6530-11d2-901f- 00c04fb951ed} - quite where these come from, I'm not sure, but the {a5dcbf10-... one matches one posted to the list, so it must be wavefinder specific. Hopefully, this code should now find any wavefinder driver, on any USB port ... **init_wavefinder()** - Trundles through the API calls captured by dabble. (see startup, below)

A few threads are kicked off - one to cycle the LEDS, one to control the frontend, and one to collect the main data: **lights_thread()** - updates all 3 LEDS every 100ms - takes a 12 bit value, 0x3ff = dim, 0x001=bright **tune_thread()** - This is where things get serious... A few commands are issued, then 0x800 bytes are returned from wavefinder. This data appears to be the raw output from the ADC in the tuner frontend - plot it, and you'll see. The VIADAB code, then performs a series of DSP functions - applies the Hamming window; fft's etc. I guess this is the tuner synchronisation, and feedback from these calculations gets fed back to keep the tuner locked. It also seems to calulate its BER from this data. This is what we need to understand more to control the fine tuning ourselves. **data_thread()** - This is also quite serious ... This thread is kept in sync with the tuner thread, and returns blocks of data (0x200 bytes), with a big discontinuity at 0x180 bytes. My guess is that this is the output of the IFFT performed by the DSP chip. (more detail based on wfic) The whole data stream is not returned - so the parameters sent each time round the loop by the tune_thread, must select which what services are being received. Since no sensible data is being sent back to wavefinder in wboot, the data_thread is likely to return garbage. But, if you run dabble, you get the same 2 distinct data streams.

wfic

Extract FIC data from Dabble4 logs, for later processing with FICDEC.

FICDEC

FICDEC

Quick and dirty DSP dissassembler

A partial DSP dissassembler for the TI 5402 DSP code has been written, but further work on this is not expected to be fruitful.

Specifications

- ETS 300 401 (AKA "radio broadcasting systems; digital audio broadcasting (DAB) to mobile, portable and fixed receivers").

(add links to above)

USB Protocol

```
Commands are sent using dwIOControlCode = 0x80002018
Data is retrieved using dwIOControlCode = 0x22000c, 0x220010, 0x2200014

USB Setup Data
    bmRequestType = 0x40 - Write
    bmRequestType = 0xc0 - Read

bRequest = 1        command/data to DSP_A
    wValue = 0x0000
    wIndex = 0x0080
    wLength = 0x40
    *data = <data>

bRequest = 2        command/data to DSP_B
    wValue = 0x0000
    wIndex = 0x0080
    wLength = 0x40
    *data = <data>

bRequest = 3        command/data to SR11R
    wValue = register address, e.g. red LED = 0xc0f4
    wIndex = register value
    wLength = 0x04,
    *data = <addr> <value>

bRequest = 4        Frequency Synthesiser
    wValue = 0x0000
    wIndex = 0x0000
    wLength = 0x0c
    *data = <data>

bRequest = 5        AFC, Symbol Selection
    wValue = 0x0000
    wIndex = 0x0000
    wLength = 0x20
    *data = <data>
```

Startup

```
(skip select_config stuff)
(skip first vendor_device packet, 64 bytes)

; initialise everything to zero
e6 c0 00 00 : 0xc0e6 : PWM control reg
e8 c0 ff 03 : 0xc0e8 : Max counter reg
ea c0 00 00 : 0xc0ea : PWM start ch.0 reg (0)
ec c0 00 00 : 0xc0ec : PWM stop  ch.0 reg (0)
ee c0 00 00 : 0xc0ee : PWM start ch.1 reg (0)
f0 c0 00 00 : 0xc0f0 : PWM stop  ch.1 reg (0)
fa c0 ff 03 : 0xc0fa : PWM cycle count - 1 (1024)
f2 c0 00 00 : 0xc0f2 : PWM start ch.2 reg (0)
f4 c0 00 00 : 0xc0f4 : PWM stop  ch.2 reg (0)
f6 c0 00 00 : 0xc0f6 : PWM start ch.3 reg (0)
f8 c0 00 00 : 0xc0f8 : PWM stop  ch.3 reg (0)

; initialise PWM counter registers
ea c0 00 00 : 0xc0ea : PWM start ch.0 reg (0)
ec c0 ff 02 : 0xc0ec : PWM stop  ch.0 reg (0x02ff)
f0 c0 ff 02 : 0xc0f0 : PWM stop  ch.1 reg (0x02ff)
f4 c0 ff 03 : 0xc0f4 : PWM stop  ch.2 reg (0x03ff)
f8 c0 ff 03 : 0xc0f8 : PWM stop  ch.3 reg (0x03ff)
f0 c0 ff 03 : 0xc0f0 : PWM stop  ch.1 reg (0x03ff)

; initialise PWM control
e6 c0 0f 80 : 0xc0e6 : PWM control register


; initialise frequency synthesiser????
; or these might be the wLength=0x0c ones, as these occur just after dabble prints out "About to tune ..."
28 c0 e0 3d : 0xc028 : I/O control register 1
20 c1 00 00 : ??????
20 c1 ff ff : ??????
24 c0 00 38 : 0xc024 : Output control register 1
1e c0 00 00 : 0xc01e : Output data register 0
24 c0 00 30 : 0xc024 : Output control register 1
24 c0 00 38 : 0xc024 : Output control register 1
14 c1 e0 00 00 00 : ??????
. . .
. . .
. . .
```

PWM Control Register = 0x800f = b1000 0000 0000 1111 which basically means it's set to begin operation, at 48MHz, continuous repeat (rather than one-shot), active low, enabled.

LEDs

The LEDs are controlled via the SL11R's built-in PWM controlled pins (of which there are four, though only three are used). The PWM pins are setup during initialisation (see above).

Colours are sent via the commands (or similar):

```
f4 c0 xx 00 (for red)
f8 c0 xx 00 (for green)
f0 c0 xx 00 (for blue)
```

with xx being the intensity (or at least the value passed to the command line of wlights). (can probably add more here, we know a lot more about the leds than this now)

Tuning Datastream

The frequency is set by a set of 6 'req=4' commands. The first five always seem to be the same; the last one varies dependant on the frequency. The wtune code just calculates the value for the last command.

```
e.g.
About to tune radio frequency to 225.648000
req=04, val=0000, idx=0000, len=000c  10 08 10 00 16 00 00 00 00 00 00 10
req=04, val=0000, idx=0000, len=000c  22 63 34 00 16 00 00 00 00 00 00 10
req=04, val=0000, idx=0000, len=000c  10 08 00 00 16 00 00 00 00 00 00 10
req=04, val=0000, idx=0000, len=000c  82 02 20 00 16 00 00 00 00 00 00 10
req=04, val=0000, idx=0000, len=000c  11 80 00 00 10 00 01 00 00 00 00 10
req=04, val=0000, idx=0000, len=000c  04 a2 03 00 13 00 01 00 00 00 00 10
                                      ^         ^
                                      \---------/

-> tune(225.648000), x1=1022e, x2=3a204
the 4 bytes calculated are 0x04, 0xa2, 0x03, 0x00
```

These values are presumably sent the to frequency synth. PLL (LMX2331U); but I've not tried to figure out the actual meanings of the parameters.

I've also been monitoring the 'req=5' commands that are sent by the tune_thread. e.g. cmd=5, value=007f, index=7fff ffff ffff fff8 0000 0000 0000 0000 0000 0000 ff00 056c 0011 0000 000f 0000

The 1st 4 words (0x7fff ffff ffff fff8) may control which symbols are transferred, but normally I see these values. Sometimes, if the tuner cannot lock (bad freq, or no reception), I see something like (1111 1111 1111 1111) which possibly resets the null symbol synchroniser. The last but few words (056c, 0011) are probably the AFC. If there is no reception or lock then these values are constant. If it is tuned, then they vary slowly, presumably causing the AFC circuit to adjust the frequency to hold the tuner in lock. - onward and upwards ...

The wavefinder return a block of 0x20c bytes per symbol received. Not all symbols per frame are passed accross the USB interface - only those selected by a 'req=5' command (precise format of this command is still TBD).

Automatic Frequency Control

The biggest missing bit is the AFC control. I figured out how to send the required frequency to the frequency synthesiser, but to complete the tuning, the frequency must be phase locked fairly precisely in order to ensure that each of the carriers end up in the middle of the FFT bin. This is a 2 stage process. The acquisistion phase achieves a course lock, by performing a sequence of correlations of the received phase reference symbol against a locally constructed phase reference symbool. By sliding the local symbol, and performing a correlation for each position, a course offset value can be determined. I can do the correlation, but I don't yet know how to feed the result back to the wavefinder. The acquisition phase can reduce the phase offset (too many phases here ...) to within 1 carrier (1 FFT bin). Once this phase offset has been reduced, the system enters a tracking phase, to provide fine adjustment. This seems to invlove a further correlation, and then uses a moving average to further adjust the frequecy. This process should be able to track any frequency drift, and reduce the phase error to \< 1 carrier - i.e centre the sample point to the middle of the carrier. As with the acquisition, I do not as yet have a complete understanding of how the analysis is converted to the value that is sent back to the wavefinder.

Main datastream - processing data blocks

My current thought is that the DSP may also do the demodulation as well as the FFT. This is mainly based on the amount of data that is returned. For each symbol, we get back 1 block of data (0x200) bytes, of which the last 0x80 bytes is a replica of the first 0x80 bytes of a previous symbol - this must be the effect of the guard band. That leaves 0x180 bytes per symbol. A symbol is based on 1536 (0x600) carriers; so that only allows 2 bits per carrier. So, either the data is 2 bits per fft bin, or the fft output has been further processed. The header for each data block indicates which symbol it is for. Symbol 0 (the NULL symbol) seems to be returned for all frames. Sometimes I also get 0x12-0x3d; somtimes I only get 2,3,4. 2,3,4 are the FIC; higher numbers are the MSC.

The data returned appears to consist of a 12 byte header that identifies the symbol number, then 0x180 bytes of 'useful' data, then a further 0x80 bytes of 'useless' data (which is presumably due to the guard band).

```
    byte 0 : 0x0c
    byte 1 : 0x62
    byte 2 : <symbol>
    byte 3 : <frame number, 00-1f>
    byte 4 : 0xb9 / 0xbd / 0xbe / 0xbf
    byte 5 : 0x20 / 0x10
    byte 6 : 0x01
    byte 7 : 0x00
    byte 8 : 0x80
    byte 9 : 0x01
    byte 10 : 0x00
    byte 11 : 0x00 / 0x80
```

The 0x180 bytes of symbol data is the result of some processing by the DSPs. The DSPs take in data from the ADC, perform a 2048 point FFT, then perform differential demodulation. The data transfered over the USB is the resulting OFDM complex symbols. These are transferred as 16 bit words. Care needs to be take to process this data in the correct order - aren't big/little endian issues a right pain...

```
e.g for the returned data : 0c 62 02 00 b9 20 01 00 80 01 00 00 6e 12 32 ...
                            <-----    12 byte header     -----> <----- symbol data ...


    6e 12 - 0 1 1 0 1 0 0 0    0 0 0 1 0 0 1 0
                                       | | | |
                                       | | | \__  real QPSK symbol 0
                                       | | \____  imag QPSK symbol 0
                                       | \______  real QPSK symbol 1
                                       \________  imag QPSK symbol 1
```

The symbol data must then be processed accroding to EN 300-401. These steps are:

- **Frequency Deinterleaving** - Each of the 1536 QPSK complex symbols is transmitted on a different carrier. Theses are reordered before transmission. This process needs to be reversed.
- **Symbol Demapping** - The 1536 QPSK symbols are derived from 3072 OFDM symbols. This process is reversed here.
- **Depuncturing** - Extra zero's are added to the data stream according to the puncturing pattern.
- **Viterbi Decoding** - The Viterbi decoder attempts to correct any errors in the data stream, using the redundancy that was added before transmission. Approx. 3 x more data is transmitted than is required.
- **Descrambling** - The data is scrambled to produce a similar number of 1's and 0's.

Command list

(perhaps we should put together a full list of USB commands and packets and what they do)

Other links

GNU Radio

- GNU Radio is a project to write a Software Defined Radio (SDR). Basically, they have hooked up a cable-TV downconverter to a wideband ADC (a $1000 PCI-card...) and are then trying to do stuff with the baseband signal in software.
- DAB ETSI Standards
- DAB Standards
- TPEG Specifications
- Good DAB Articles
- BBC MOT Webpages
- BBC TPEG Webpages
- VIADAB Specification

- 2 API.pdf VIADAB 2 Specification
- DABUSB
- Wavefinder FAQ
- DABbar
- Yahoo DABUSB
- alt.radio.digital
- Rhode & Scwartz DAB COFDM Generator
- (many more on DABUSB Yahoo group bookmarks)

Original editor: Matthew Burnham Last updated 31 May 2009 by md84419

Categories:

- OMAP
- BeagleBoard

> From: eLinux.org

ZipIt

Contents

Introduction

The **ZipIt** Wireless Instant Messenger is made by Aeronix and imported from China. The official site is at ZipItWireless.com. The **ZipIt** runs a custom IM app running under linux. The GPL/LGPL source code is available or documented. However, the object file for Zipit binary is not available for re-linking so they are in violation of the LGPL license for glibc.

There is an active community of users working to extend the functionality of the **ZipIt**. This group maintains this wiki and a Yahoo! group.

- **Technical details** of the **ZipIt** (hardware & software) can be found on the ZipIt Tech Details page.
- Details on **modifications** you can make to the Zipit (backlight, flash menory, ports) Zipit Hardware Mods
- Alternative **software** created by users is available ZipIt Software

The **ZipIt** might be purchased from Radio Shack (The Source) for $99 or amazon and other online retailers.

How To...

- A frequently asked questions list is available at ZipIt FAQ
- Add hardware features to your Zipit (e.g. backlight, clock, MMC/SD memory, etc.) Zipit Hardware Mods
- How to perform the 5-wire and 3-wire serial mods so you can alter the Zipit firmware: Zipit Serial Mod original website: http://aibohack.com/zipit/serial.htm
- How to flash new firmware via serial connection ZipIt Serial Flash
- How to flash new firmware with no hardware modification over WiFi. (only works on Zipits with firmware versions before 2.01) ZipIt WiFi Flash
- How to setup an NFS server for windows to update to any other distro for the zipit: ZipIt Winxp NFS]
- How to setup Adams software ZipIt Adam HOWTO
- How to compile new software for use on **ZipIt**: ZipIt Compile
- Tutorial on developing applications for **ZipIt**: ZipIt Developer Tutorial
- Sample program to write to framebuffer: ZipIt Framebuffer Example
- How to connect to WiFI networks from Linux command line: ZipIt WiFi Connect
- Information about Audio on the **ZipIt**: ZipIt Audio
- What to do when you use the wrong power supply www.openhardware.net
- Linux zipit tools zipit tools

Hardware Overview

- Cirrus EP7312-CR-90 (ARM720T CPU running at up to 90MHz)

- 2M flash - MX 29LV160ATxBC-70
- 16M SDRAM - Hynix HY5V26D
- 320x240x4 LCD (16 level gray scale)
- Agere WiFi chip - WL600114LY
- Wolfson Micro WM8751L Stereo DAC
- LPC915 (8 bit uP)

http://www.elkgrovewireless.com/zipit/zipit.jpg

Zipit Comments Not On Yahoo

Additional Zipit Notes

External Articles & Reviews

http://www.linuxdevices.com/articles/AT8107883197.html http://hardware.slashdot.org/article.pl?sid=05/07/17/1416202&tid=100 http://www.elkgrovewireless.com/zipit/

Project Updates

July 2005

- Greatly expanded information on installing the BURN3 and OpenZipIt firmware, especially from Linux.
- Many tech details have been moved to the ZipIt Tech Details page in order to simplify this main page.

June 2005

- Added two How-Tos: ZipIt WiFi Flash and ZipIt Compile
- A software-only firmware modification is outlined in the former link.

March 2005

- Keep checking the Yahoo BBS for the latest current status
- The core Linux components are now replaceable (including newer BusyBox, telnetd etc). Also a way of reflashing a non-solder modified device has been found.
- Eventually a software-only SDK will be released (soldering not necessary if you are careful with subsequent reflashing steps)

Orphaned Zipit pages

The following Zipit pages were found orphaned. I've added links here. Someone more knowlegeable please move them to a more proper location (or remove them if they are not needed any more)

ZipItV2 JTAG ZipItV2 UART ZipIt NFS WL-HDD25 ZipIt To Do List Zipit Scan Delta_F

Category:

- Zipit

> From: eLinux.org

Development Platforms

Contents

Popular Devices

- Via APC 8750
- Raspberry Pi from the RaspberryPi foundation based on on BCM2835 from Broadcom
- CraneBoard from Mistral Solutions based on AM3517 from Texas Instruments
- AM/DM37x EVM from Mistral Solutions based on AM/DM35x processors from Texas Instruments
- BeagleBoard Texas Instruments OMAP3(Cortex-A8) based
- Devkit8000 Texas Instruments OMAP3530(Cortex-A8) based board
- Devkit8500D TI DM3730 ARM Cortex-A8 based board
- Devkit7000 Samsung S5PV210 ARM Cortex-A8 based board
- Hawkboard Texas Instruments OMAP L138(ARM9 and C674X Floating Point DSP) based
- Hammer_Board ARM9 based Samsung S3C2410
- Jetson TK1 NVIDIA Tegra K1 quad-core Cortex-A15 CPU + **192-core Kepler GPU** mobile super-computer. With U-Boot + Nouveau it becomes the first **100% fully open-source** GPU-accelerated Linux development board!
- LeopardBoard Texas Instruments TMS320DM355 based
- Opensourcemid K7 MID OMAP3530 tablet from OpenSourceMID.org
- OMAP3 EVM from Mistral Solutions
- Odroid Samsung S5PC100 (Cortex-A8) based 'Hackable Android handheld game device'
- (A beagle clone)
- Snowball SDK & PDK (Dual Cortex A9 + Mali400)
- Lionboard Texas Instrument DM368 SODIMM module (ARM9 + video core)
- MYD-SAMA5D3X board for Atmel SAMA5D3 (ARM Cortex-A5) designed by MYIR
- MYD-AM335X board for TI AM335X (ARM Cortex-A8) designed by MYIR
- MYD-IMX28X board for Freescale i.MX28 (ARM9) designed by MYIR
- MYD-SAM9X5 board for Atmel AT91SAM9G15/G25/G35/X25/X35 (ARM9) designed by MYIR
- MYD-SAM9X5-V2 board for Atmel AT91SAM9G15/G25/G35/X25/X35 (ARM9) designed by MYIR

ARM

- Forlinx Embedded ARM series boards
  - FL2440 boards based on Samsung ARM9 S3C2440
  - FL2416 boards based on Samsung ARM9 S3C2416
  - OK6410-A boards based on Samsung ARM11 S3C6410

  - OK6410-B boards based on Samsung ARM11 S3C6410
  - OK210 boards based on Samsung Cortex-A8 S5pv210
  - OK210-A boards based on Samsung Cortex-A8 S5pv210
  - OK335xD boards based on TI Sitara AM335x
  - OK335xS boards based on TI Sitara AM335x
  - OK335xS-II boards based on TI Sitara AM335x
- GNUBLIN GNUBLIN board

- ARM cortex A8 board 1GHz 512M DDR3 Samsung S5PV210 based
- Devkit8000 OMAP3530 based board from Embest
- Devkit8500D TI DM3730 based board
- Devkit7000 Samsung S5PV210 based board
- CraneBoard from Mistral Solutions
- PandaBoard
- AM/DM37x EVM from Mistral Solutions
- Digi JumpStart Kits - http://www.digi.com/products/embeddedsolutions/softwareservices/digiembeddedlinux.jsp
- SheevaPlug
- GuruPlug - http://hackaday.com/2010/02/08/guruplug-the-next-generation-of-sheevaplug/
- ARM Integrator
- OSK - OMAP Starter Kit
- GAO Engineering Inc. - http://www.gaoengineering.com
- DaVinci DVEVM Evaluation module - http://www.spectrumdigital.com/
- ITSY
- LART Project
- Hammer_Board
- Simtec Electronics - http://www.simtec.co.uk/
- Open AT91RM9200 Evaluation Board - http://wiki.emqbit.com/free-ecb-at91
- BeagleBoard
- ODROID Samsung Exynos based development boards and tablets
- Balloonboard
- KB9202 - http://www.kwikbyte.com/KB9202.html
- Luminary Micro's **LM3S6965** is an ARM Cortex M3 MCU. There is an inexpensive development board available for it called the **LM3S6965 Ethernet Evaluation Kit**, which is available from Mouser and others for around $69.00 USD.
- TechnologicSystems
- OMAP3 EVM from Mistral Solutions
- NaviEngine NEC ARM11MPCore (4 x ARM11) based
- Freescale i.MX based Armadeus APF boards
- Zoom OMAP34x & 36x Development Kit - http://omapzoom.org
- Tegra2
- Arm11 development board
- Snowball SDK & PDK (Dual Cortex A9 + Mali 400)
- Raspberry Pi
- Freescale IMX53QSB [1]
- Calao Atmel AT91 development board [2]
  - USB A9260
  - USB A9263
  - USB A9G20
  - TNY A9263
  - TNY A9G20
  - QIL-A9260
- FriendlyArm boards:
  - Mini2440 | S3C2440 ARM9 Board
  - Tiny6410 | S3C6410 ARM11 Board

- Basi and Dingo DaVinci dm365 boards
- Gumstix Overo
- Embedded Open Modular Architecture/EOMA-68
- List of OMAP boards
- Dragonboard
- WandBoard
- Colibri Evaluation Board for Nvidia Tegra T20, T30 and Freescale iMX6, VF50, VF61 by Toradex Switzerland and Seattle, WA
- Apalis Evaluation Board for Nvidia Tegra T30 and Freescale iMX6 by Toradex Switzerland and Seattle, WA
- Ixora Carrier Board for Nvidia Tegra T30 and Freescale iMX6 by Toradex Switzerland and Seattle, WA
- Viola Carrier Board for Nvidia Tegra T20, T30 and Freescale VF50, VF61, iMX6 by Toradex Switzerland and Seattle, WA
- Iris Carrier Board for Nvidia Tegra T20 and T30, Freescale iMX6, VF50, VF61 and Intel/Marvell PXA270, PXA310, PXA320 by Toradex Switzerland and Seattle, WA
- Orchid Carrier Board for Intel/Marvell PXA270, PXA310, PXA320, Nvidia Tegra T20 and T30 and Freescale iMX6, VF50, VF61 by Toradex Switzerland and Seattle, WA
- PengPod
- A13 OLinuXino-MICRO Allwinner A13 development board
- Boundary Devices i.MX6 boards
  - Nitrogen6x
  - Nitrogen6 Lite
  - SABRE Lite
- SolidRun HummingBoard, i.MX6 with Raspberry Pi form factor, connectors and pinouts
- MYIR's ARM boards
  - MYD-AM335X | TI AM335x ARM Cortex-A8 Development Boards
  - MYD-IMX28X | Freescale IMX28X ARM9 Development Boards
  - MYD-SAMA5D3X | Atmel SAMA5D3 ARM Cortex-A5 Development Boards
  - MYD-SAM9X5 | Atmel AT91SAM9X5 ARM9 Development Boards
  - MYD-SAM9X5-V2 | Atmel AT91SAM9X5 ARM9 Development Boards
  - MYC-SAMA5D3X | Atmel SAMA5D3 ARM Cortex-A5 CPU Modules
  - MYC-SAM9X5 | Atmel AT91SAM9X5 ARM9 CPU Modules
  - MYC-SAM9X5-V2 | Atmel AT91SAM9X5 ARM9 CPU Modules
  - MYS-SAM9X5 | Atmel AT91SAM9X5 ARM9 Single Board Computers
  - MYS-SAM9G45 | Atmel AT91SAM9G45 ARM9 Single Board Computers
  - MYD-LPC435X | NXP LPC4350/4357 ARM Cortex-M4/M0 Development Board
  - MYD-LPC185X | NXP LPC1850/1857 ARM Cortex-M3 Development Board
  - MYD-LPC1788 | NXP LPC1788 ARM Cortex-M3 Development Board
- Atmel Xplained fast prototyping boards
  - Atmel SAMA5D3 Xplained
  - Atmel SAMA5D4 Xplained

AVR32

- ATNGW100 Network Gateway Kit http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4102
- AVR dev kit board
- AVR JTAG Emulator

Blackfin

- ADI Blackfin Boards
- ADI Boards

- Add on cards for ADI Boards
- Everyone else
- Demo video
- ADI JTAG Emulators

MIPS

- mips-1 little endian - Flameman/routerboard-rb532
- Ben NanoNote
- MIPS Creator CI20 development board
- Opendreambox DVB/PVR set-top-box family http://www.opendreambox.org
- MiniEMBWiFi

PowerPC

- Walnut (405GP) - http://amcc.com/Embedded/Downloads/download.html?cat=1&family=2
- Dht-Walnut (405GP) - http://www.elinux.org/Flameman/dht-walnut
- Walnut (405GP) - http://www.elinux.org/Flameman/walnut
- Ebony (440GP) - http://www.elinux.org/Flameman ask flameman about
- sandpoint (7410) - http://www.elinux.org/Flameman/sandpoint3
- Kuro Box-HG (MPC4281) - http://www.kurobox.com/mwiki/index.php/Kurobox/Kurobox-HG_Main_Page
- Efika5200 (MPC5200) - http://www.powerdeveloper.org/program/efika/accepted

SuperH

- Sega
  - Dreamcast(SH7091) - Linux-SH Dreamcast Note that only machines with a a model number that begin with MIL-CD are supported. It is highly suggested to use a BBA for communication with the Dreamcast.
- Hitachi ULSI Systems
  - MS7206SE01 (SH72060 Solution Engine)
  - MS7750SE01 (SH7750(sh4) Solution Engine)
  - MS7709SE01 (SH7709(sh3) Solution Engine)
- SuperH, Inc.
  - MicroDev
- HP Jornada
  - 525 (SH7709 (sh3))
  - 548 (SH7709A (sh3))
  - 620LX (SH7709 (sh3))
  - 660LX (SH7709 (sh3))
  - 680 (SH7709A (sh3))
  - 690 (SH7709A (sh3))
- Renesas Technology Corp.
  - RTS7751R2D CE Linux Forum（CELF）Compliant Evaluation Board
- Renesas Europe/MPC Data Limited
  - EDOSK7705 (SH7705 sh3)
  - EDOSK7760 (SH7760 sh4)
  - EDOSK7751R (SH7751R sh4)
  - SH7751R SystemH (SH7751R sh4)
- CQ Publishing Co.，Ltd.
  - CQ RISC Evaluation Kit(CqREEK)/SH4-PCI with Linux

- Kyoto Microcomputer Co., Ltd. (KMC or KμC)
  - Solution Platform KZP-01 (KZP-01[Mainboard] + KZ-SH4RPCI-01[SH4 CPU Board])
- Silicon Linux Co,. Ltd.
  - CAT760 (SH7760)
  - CAT709 (SH7709S)
  - CAT68701 (SH7708R For A-one CATBUS[Designed for 68000 board] compliant)
- Daisen Electronic Industrial Co., Ltd.
  - SH2000 (SH7709A 118MHz)
  - SH2002 (SH7709S 200MHz)
  - SH-500 (SH7709S 118MHz)
  - SH-1000 (SH7709S 133MHz)
  - SH-2004 (SH7750R 240MHz)
- [IO-DATA DEVICE, Inc.(Network Attached Storage NAS Series)]
  - LAN-iCN (NAS Adapter for IODATA HDD with "i-connect" Interface)
  - LAN-iCN2 (NAS Adapter for IODATA HDD with "i-connect" Interface)
  - LANDISK (SH4-266MHz[FSB133MHz] RAM64MB UDMA133 USB x2 10/100Base-T)
  - HDL-xxxU (LANDISK Series NAS Standard Model)
  - HDL-xxxUR(LANDISK with RICOH IPSiO G series print monitor for Windows support)
  - HDL-WxxxU(LANDISK with wide body & twin drive support for Heavy storage or RAID1)
  - HDL-AV250(LANDISK with Home Network DLNA guideline support)
  - LANTank(LANDISK kit SuperTank(CHALLENGER) Series)
  - HDL-WxxxU based twin drive bulk NAS kit. LANTank have a special feature witch supported network media server(cf. iTunes etc..).
- TOWA MECCS CORPORATION
  - TMM1000 (SH7709)
  - TMM1100 (SH7727)
  - TMM1200 (SH7727)
- Sophia Systems
  - Sophia SH7709A Evaluation Board
  - Sophia SH7750 Evaluation Board
  - Sophia SH7751 Evaluation Board
- MovingEye Inc.
  - A3pci7003 (Using SH7750/ART-Linux [Linux with Realtime Extension])
- AlphaProject Co., Ltd.
  - MS104-SH4 (SH7750R/PC104(Embedded ISA Bus) with apLinux)
- Interface Corporation.
  - MPC-SH02 (SH7750S: ATX Motherboard Style)
  - PCI-SH02xx (SH7750S: PCI-CARD Style)
- TAC Inc.
  - T-SH7706LAN another name "Mitsuiwa SH3 board" ["SH-MIN"] (SH7706A/128MHz Flash512KB SDRAM 8MB 10BASE-T)
- SecureComputing/SnapGear (older products, check ebay etc, all can netboot and have a debug header)
  - SG530 (SH7751@166MHz RAM16MB FLASH4MB 2x10/100 1xSerial)
  - SG550 (SH7751@166MHz RAM16MB FLASH8MB 2x10/100 1xSerial)
  - SG570 (SH7751R@240MHz RAM16MB FLASH8MB 3x10/100 1xSerial)
  - SG575 (SH7751R@240MHz RAM64MB FLASH16MB 3x10/100 1xSerial)
  - SG630 (SH7751@166MHz PCI NIC card RAM16MB FLASH4MB 1x10/100 1xSerial-header)
  - SG635 (SH7751R@240MHz PCI NIC card RAM16MB FLASH16MB 1x10/100 1xSerial-header)

i386 and compatible

- CR48 Google netbook

- Bifferboard

Unclassified

- http://www.mikrotik.com/
- http://www.routerboard.com/
- http://www.cuwireless.net/
- http://leaf.sourceforge.net/
- http://leaf.sourceforge.net/mod.php?mod=userpage&menu=908&page_id=27
- http://www.myirtech.com/
- http://www.1st-safety.com/arm/Tiny6410/
- StalkerBoard
- SBC3530
- SBC8100
- SFFSDR
- SOM1808
- MINI2440v2_developmentboard
- Launchpad
- Micro2440
- Mini210
- Tiny210

Categories:

- Hardware
- Development Boards

> From: eLinux.org

A13 OLinuXino-MICRO

Contents

JTAG support

The A13 OLinuXino-MICRO has JTAG support brought out to an unpopulated 6-pin header next to the headphone jack. Solder a piece of header strip and it's ready to go.

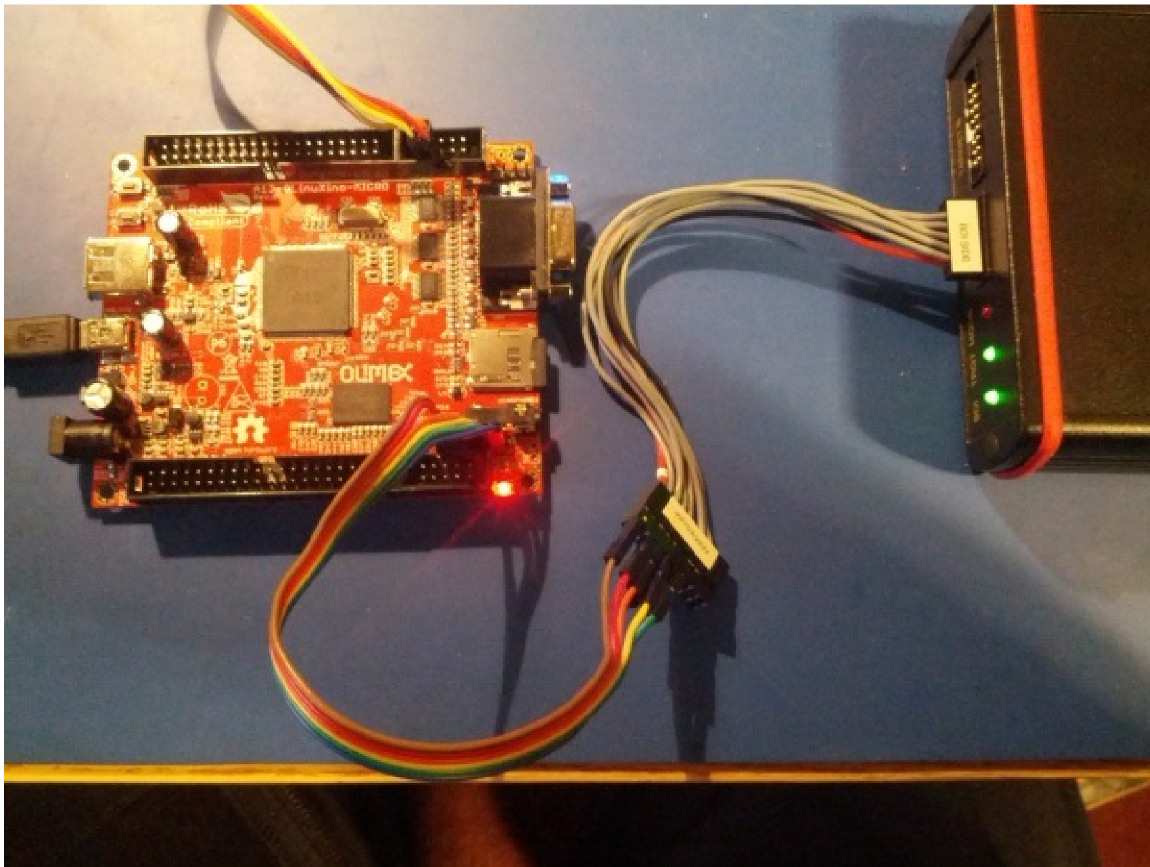
Cabling

The table below shows the pinout for the proprietary JTAG header on the board. All pins are clearly silkscreened on the bottom of the board.

A13 OLinuXino JTAG	Pin	JTAG Signal	ARM 20-pin JTAG	ARM 14-pin
GND	1	GND	10	10
TDO	2	TDO	13	11
TCK	3	TCK	9	9
TMS	4	TMS	7	7
TDI	5	TDI	5	5
3.3V	6	3.3V	1	1

BDI-2000/3000

The A13 OLinuXino-MICRO BDI config file can be used with a BDI configured with the ARM11/Cortex-An firmware and the standard BDI ARM/Xscale 20-pin JTAG cable.

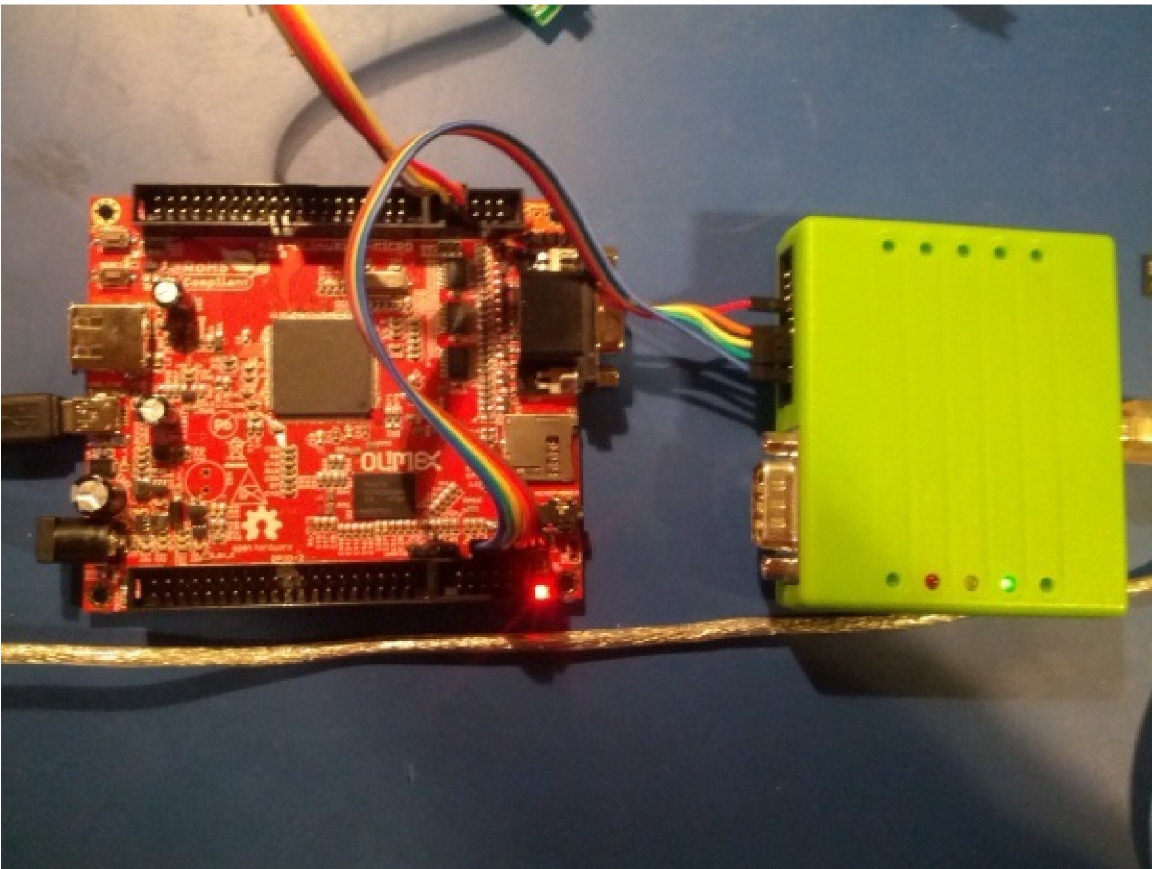


Flyswatter 2 / OpenOCD

The A13 OLinuXino OpenOCD config file can be used with a Flyswatter, OpenOCD, and the ARM 20-pin JTAG on the Flyswatter 2.

Usage:

```
openocd -f interface/flyswatter2.cfg -f /tmp/a13-olinuxino.cfg
```



Boot Layout

The device doesn't have internal storage and boots from the SD-card. The second and third stage loader (SPL and U-Boot) are loaded from the start of the MMC (not on a file system). The default configuration is to create two partitions the first is a FAT formatted one where u-boot will load files from and the second one probably a ext2 partition. But given the bootloaders are not on the FAT you need to reserve some space at the start (I leave 2048 512 byte blokcs at the start as this is the default fdisk offers).

The layout looks something like this:

NAME	start block	size
MBR	0	1 block
sunxi-spl.bin	16	20K
u-boot.bin	64	170K
FAT	2048	20M
EXT2	---	Rest

Categories:

- Flyswatter
- HOWTOs

> From: eLinux.org

Real6410 Single Board Computer

(Redirected from Arm11 development board)

Contents

Real6410

The Real6410 Single Board Computer is a high-performance controller board introduced by CoreWind. It is designed based on the Core6410 processor card which integrates an S3C6410 microcontroller, 256MByte mDDR SDRAM, 1GByte Nand Flash, RTC, Audio and net on board. It is connected with Real6410 expansion board through 170pin expansion interfaces（QFP package）.

In addition to those features provided by the CPU board CoreWind, the expansion board has exposed many of other features of the S3C6410. It has integrated RS232, USB, Ethernet, WiFi, GPS, GPRS Audio In/Out, Keyboard, LCD, CVBS、TV out, camera in, SD card and more other functions on board. So many hardware resources provided by the expansion board, it becomes a solid reference board for customer design.

CoreWind also offers a complete software development package to customers. The board supports linux 2.6.28, Android2.1 and WindowsCE 6.0 operating system and is provided with complete basic drivers which enable a quick channel to evaluate the Samsung S3C6410 processor and customize application software. It would be an ideal development platform for multimedia and communication applications.

Highlights

- S3C6410? development board
- ARM11 Samsung S3C6410?, ARM1176JZF-S, up to 667MHz
- 256MByte MoblieDDR, 266MHz
- 1GByte NAND Flash
- LCD/Touch Screen, CVBS/TV, Audio support
- SD card, USB Host/OTG support
- Ethernet, serial port,
- GPRS/WIFI/GPS/Camera support(option)

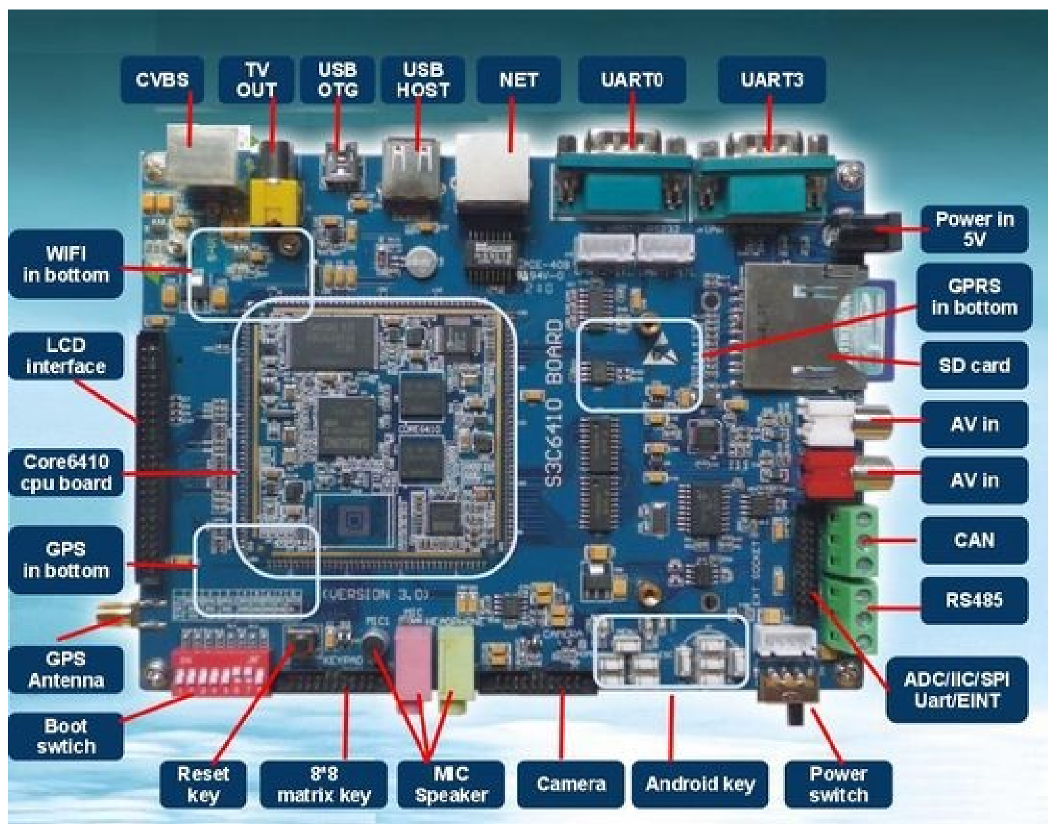
- Linux 2.6.28, WinCE6.0 support
- Android2.1 support
- All source code provide

[More product Details >>]

Developers' Resources

	Developers' Resources Join and share full development resources of Real6410 Start Here! >> Linux development >> WinCE development >> Android 2.1 >>			**Hardware Information** Major component information available Real6410 board >>]			**Market Partner** Real6410 product Marketplace Go >>

HardWare



• Mechanical Parameters ◦ Dimensions:168.3 * 124 mm ◦ Input Voltage: +5V ◦ Temperature Range: 0 ℃ ~ 70 ℃ ◦ Humidity Range: 20% ~ 90% • Audio/Video Interfaces ◦ A TFT LCD interface,up to 2048*2048 ◦ A CVBS 、TV out interface ◦ A camera input interface ◦ A audio I/O interface ◦ A 1.5W Speaker ◦ 4 line Touch Screen • Input Interface ◦ 10 phone button ◦ 8 x 8 Matrix Key interface ◦ One Reset button ◦ Boot mode switch	• Data Transfer Interface • Serial port: ◦ Two 3-line serial port, RS232 voltage • USB port: ◦ 1 x USB OTG 2.0 ◦ 1 x USB Host 1.1 • SD card slot: ◦ 1 channel SD card slot • Ethernet: DM9000AEP 10/100Mbps, RJ45 connector • 6 LEDs (programmable status LEDs, on CPU board) • extend interface: SPI, IIC, ADC, EINT • GPRS Module ◦ Module name: GPRS SIM300 ◦ Link method: Serial port • WIFI Module ◦ Module name: AW-GM320 (azurewave company) ◦ Link method: SDIO • GPS module ◦ Module name: SIRFIII GPS EB818 ◦ Link method: Serial port • Camera module ◦ Module name: OV9650

Software

WinCE 6.0

- Boot loader
  - Version: Stepldr and EBOOT(provide Source code)
  - Function: support download and update system by SD and USB
- WinCE NK
  - Version：Wince6.0

  - Function: HIVE register support, BINFS support, 256M memory manage, SLEEP
- Device Drivers
  - TFT LCD/Touchscreen, TV out, Audio I/O, MMC/SD card, NET, USB host, USB OTG, Serial port
  - watchdog, RTC, Power Manage, button, LED,
  - WIFI, GPS, GPRS, Camera
- System characteristic
  - Windows Media Player 9.0 (supports MP3, MPEG2, MPEG4, WMV, WAV and so on).
  - Picture explorer, wordpad.
  - IE6 explorer.

Linux 2.6.28

- Boot loader
  - verison: s3c-u-boot-1.1.6
  - Function: support boot and update system by SD card and USB
- Linux kernel
  - verison: s3c-Linux-2.6.28.4
  - Compile: arm-none-linux-gnueabi-4.3.2
  - Function: support MFC, Jpeg encode, 2D/3D
- Device Driver
  - TFT LCD/Touchscreen, Audio I/O, MMC/SD card, NET, USB host, USB OTG, Serial port

watchdog, RTC, Power Manage, matix keybad, button, i2c, spi, ADC WIFI, GPS, GPRS, Camera

- File System support
  - ubifs/yaffs2/cramfs/fat32
- Gui support
  - qtopia-2.2.0
  - QtE-4.5.2

Android 2.1

- Boot loader
  - verison: s3c-u-boot-1.1.6
  - Function: support boot and update system by SD card and USB
- Linux kernel
  - verison: s3c-Linux-2.6.28.4
  - Compile: arm-none-linux-gnueabi-4.3.2 and jdk5
  - Function: support Jpeg encode, USB adb debug, AVD support
- Device Driver
  - TFT LCD/Touchscreen, Audio OUT, MMC/SD card, NET, USB OTG, Serial port
  - watchdog, RTC, keyboard
  - Support WIFI, GPS, Camera driver
- File System support
  - Ubi filesystem
- Function Test demo
  - Ethernet
  - USB adb debug
  - Switch horizontal and vertical screen
  - Dynamic Wallpapers
  - APK program install method
  - WIFI Test, Camera test, GPS module Test

How To Purchase

The Real6410 is sold directly from CoreWind. [http://www.real6410.com/buynow]

Manuals

1) [Real6410 Hardware Development manual .pdf]

2) [Real6410 Android2.1 Development manual.pdf]

3) [Real6410 Linux Development manual.pdf]

4) [Real6410 WinCE Development manual.pdf]

Category:

- ARM Development Boards

From: eLinux.org

Armadeus APF boards

APF boards are Systems On Module designed for embedding into project needing performances, low power and versatility. Initiated by a group of embedded systems enthusiasts as a proof of concept (ARM+FPGA combination), boards are now designed, sold and supported by the ARMadeus Systems company.

Open Source Buildroot's based BSP is maintained by ARMadeus Systems itself and boards are used by the growing Armadeus Project Community.

FPGA's IP designed for the boards support OpenCores Wishbone bus.

APF51 (i.MX51 based)

	The APF51 is a reduced size processor board fitted with a 800MHz i.MX51 processor, 64 MB to 512 MB LPDDR RAM (32 bits), 256 MB to 32 GB FLASH (NAND SLC), a 10/100Mbits Ethernet port and a Xilinx **Spartan 6A FPGA**, it is easily integrated into an embedded system thanks notably to its PMIC and on board PHYs (Ethernet & USB). Click here for official APF51 site	

APF27 (i.MX27 based)

	The APF27 is a reduced size processor board fitted with a 400MHz i.MX27 processor, 64 MB to 256 MB DDR mobile RAM (32 bits), 256 MB to 512 MB FLASH (NAND 16 bits), a 10/100Mbits Ethernet port and a Xilinx (200K gate) **Spartan 3A FPGA**, it is easily integrated into an embedded system thanks notably to its regulators and level converters (RS232/USB). Click here for official APF27 site	

APF9328 (i.MXL based)

	The APF9328 is a reduced size processor board fitted with a 200MHz i.MXL processor, 8 to 32MB SDRAM, 8 to 16MB FLASH (NOR), a 10/100Mbits Ethernet port and a Xilinx (200K gate) Spartan 3 FPGA. It is easily integrable into an embedded system thanks notably to its regulators and level converters (RS232/USB). **This board is no more recommended for new projects.**	

Category:

- ARM Development Boards

> From: eLinux.org

ARM Integrator Info

Integrator Compact Platform

The Real View® Integrator(TM) AP and CP (Compact Platform) operates in conjunction with a Core Module to provide a fully integrated development platform. The Core Module FPGA implements a set of ARM PrimeCell® Peripherals and memory controllers. The PrimeCell Peripherals make use of the drivers and connectors on the Compact Platform baseboard; these include LCD and touch screen connectors, VGA connector, Multi-Media Card (MMC) interface and an audio codec interface. Ethernet connectivity is provided by a dedicated interface chip. Expansion to Logic Modules and Logic Tiles is through an AMBA AHB-Lite system bus.

- The Integrator/AP is a basic board with two UART ports, timer, RTC and NOR Flash
- The Integrator/CP can be extended to support user created IP, using standard Integrator Logic Tiles, in conjunction with an IM-LT1 Interface, or using Logic Modules.

Integrator/CP is compatible with the following Integrator Core Modules: ARM920T, ARM922T-XA10, ARM926EJ-S, ARM946E-S and ARM966E-S, ARM1026EJ-S and ARM1136J(F)-S. See the following link Integrator Core Modules for a more complete description of the board and processors.

The Integrator platforms are no longer officially supported by ARM Ltd.

Boot Loader

U-boot contains support for this board. To compile U-Boot from CVS sources you can use the following commands:

```
*make integratorcp_config
*make
```

The resulting executable can then be loaded on the board using a debugger to be run from the bootMonitor. The following page Prebuilt downloads contains some information on compiling and running Linux for this board.

Links

- Booting a recent U-Boot and kernel on ARM Integrator/AP

Category:

- Categories

> From: eLinux.org

ARM Processor

The ARM architecture (previously, the Advanced RISC Machine, and prior to that Acorn RISC Machine) is a 32-bit RISC processor architecture developed by ARM Limited that is widely used in a number of embedded designs. Because of their power saving features, ARM CPUs are dominant in the mobile electronics market, where low power consumption is a critical design goal.

Today, the ARM family accounts for approximately 75% of all embedded 32-bit RISC CPUs,making it one of the most prolific 32-bit architectures in the world. ARM CPUs are found in all corners of consumer electronics, from portable devices (PDAs, mobile phones, media players, handheld gaming units, and calculators) to computer peripherals (hard drives, desktop routers). Important branches in this family include:

- Marvell's XScale
- Texas Instruments OMAP series
- Samsung's S3C24xx series
- NXP's Bluestreak series
- Freescale's I.MX series
- Atmel
- Boardcom
- Qualcomm
- AllWinner
- RockChip

TECHNICAL INFORMATION

CPU Core	MMU/MPU	ISA
StrongArm	MMU	v4
ARM7TDMI	none	v4T
ARM7EJ-S	none	v5TEJ
ARM720T	MMU	v4T
ARM920T	MMU	v4T
ARM922T	MMU	v4T
ARM926EJ-S	MMU	v5TEJ
ARM940T	MPU	v4T
XScale	MPU	v5TE
ARM946E-S	MPU	v5TE
ARM966E-S	none	v5TE
ARM1020E	MMU	v5TE
ARM1022E	MMU	v5TE
ARM1026EJ-S	MMU+MPU	v5TE
ARM1136J-S	MMU	v6
ARM1136JF-S	MMU	v6
Cortex-M1	none	v6-M
Cortex-A8	MMU	v7-A
Cortex-R4	MPU optional	v7-R
Cortex-M3	MPU	v7-M

CPU Core	Pipeline Depth	Typical MHz
ARM7	3 stage	80
StrongArm	5 stage	133
ARM9	5 stage	150
ARM10	6 stage	260
XScale	8 stage	400
ARM11	8 stage	335

NOTE: increased pipeline length reduces the amount of work done at each stage in the pipeline, therefor enabling higher operating frequencies and performance. however, as the pipeline length increases, system latency also increases due to increased number of clock cycles needed to fill the pipeline before an instruction can be executed. an example would be an ARM920T running at 400MHz might have comparable performance to an Xscale running at 600MHz

Categories:

- ARM processors
- Processors

> From: eLinux.org

ATNGW100

The NGW100 is a quite cheap (~100USD, 50 for students) development kit for the AVR32 processor.

Features:

- AT32AP700x CPU, 140Mhz
- Input Voltage 9-15V
- SD/MMC Cardreader
- 2x Ethernet
- 1x RS232
- 3x Expansion Headers
- Device USB (offers access to the sdcard)
- 32MB Ram

Through the expansion headers it is possible to attach an lcd display to it, as well as ps/2 keyboard.

It uses U-Boot as a bootloader.

Buildroot offers out-of-the box support for this board. AVR32 support was removed in 02.2015. OpenADK can be used instead.

Some /proc files

```
~ # cat /proc/mtd
dev:    size   erasesize  name
mtd0: 00020000 00010000 "u-boot"
mtd1: 007d0000 00010000 "root"
mtd2: 00010000 00010000 "env"
mtd3: 00840000 00000420 "spi0.0-AT45DB642x"



~ # cat /proc/cpuinfo
processor       : 0
chip type       : AT32AP700x revision C
cpu arch        : AVR32B revision 1
cpu core        : AP7 revision 0
cpu MHz         : 140.000
i-cache         : 16K (4 ways x 128 sets x 32)
d-cache         : 16K (4 ways x 128 sets x 32)
features        : dsp simd ocd perfctr java
bogomips        : 281.93
```

Projects

NGW100-RTC - Add an external RTC to the NGW100

= Further reading

http://www.avrfreaks.net/wiki/index.php/Documentation:NGW

Category:

- Development Boards

> From: eLinux.org

Balloonboard

Balloon is an Open Hardware board, designed for embedding into projects needed low-power, high-performance poartable computing. It can be built in various configurations, with either FPGA for maximum welly, or CPLD for minimising power consumption. It is based around the Xscale 270.



The project has its origins with some of the people involved with making LARTs, and Balloon2 saw the light of day in in 2003, and over 5000 have been made.

Balloon3 has had a long gestation with enormous support from Cambridge University Engineering Dept. It is now available, and has a company (iEndian) formed around it which is running production, sales and support.

Specs are available on the iEndian site or on the Balloonboard Wiki.

Category:

- Hardware

> From: eLinux.org

Basi and Dingo DaVinci dm365 boards

Preliminary informations

File:Basi and dingo.pdf

Category:

- Davinci

> From: eLinux.org

Blackfin

The Blackfin processor was designed by and is manufactured by Analog Devices (Board and Chip Vendors#A). Several boards that include a Blackfin processor are available (Hardware Hacking#Blackfin).

All Blackfin processors include Debug/JTAG Interface for in-system debugging.

Blackfin processors are fast enough to support real-time H.264 video encoding.

The Blackfin processor documentation describes a Memory Management Unit (MMU), but this is not a MMU in the sense that most people expect: it has no Virtual Memory (VM) support. It does however provide all other aspects associated with your typical MMU on a variable page sized basis such as Memory Protection (read/write/execute) and caching. The documentation refers to these mappings as the Cacheability Protection Lookaside Buffer (CPLD) tables and are similar to TLBs in most other processor architectures.

As such, all current Blackfin parts can only run Linux with MMU support turned off (what people have historically thought of as "uClinux").

The processors also fully enforces privileged execution -- the classical user / supervisor permission split. The Linux kernel runs in supervisor mode where it has full access to the hardware while the Linux userspace runs in user mode and prevents tampering with hardware resources.

Further reading

- the Blackfin Linux wiki
- the Blackfin Open Source Koop

Category:

- Processors

> From: eLinux.org

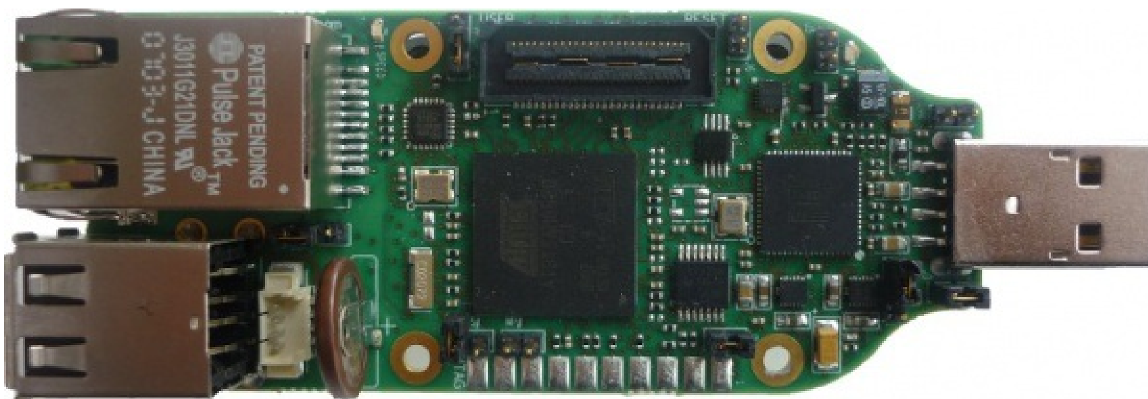
Calao Atmel AT91 development board

Calao System propose Atmel AT91 ARM development Board with small form factor USB, Tiny, QIL

Contents

USB Form Factor

USB-A9G20-LPW-C01



Booard Dimension	36 x 85 mm
Soc	ATMEL AT91SAM9G20 @ 400MHz
NAND	256MB NAND Flash (8bits),
SDRAM	64MB SDRAM (32bits @ 133MHz)
Ethernet	1x 10/100
USB Host	2x Ports
USB Device	1x Port USB (front shared with DBGU/JTAG or back)
Debug	1x JTAG & 1x DBGU over USB (FTDI FT2232D) or 1x JTAG & 1x DBGU on board
RTC	1
SD Card	1x Micro-SD (SPI under ethernet port) or 1 via Expansion port
Expansion	1x 50 pins connector (I2C, SPI, USART, SSC, MCI, ISI, TC)

AT19Bootstrap

Barebox

You can use barebox on with the following support

uart

nand

ethernet

usb host

sd card (mci)

spi

generic barebox features (menu, hush, multi-devices, etc...)

1) First you need to clone the tree

The board support is currently in the next branch

```
git clone git://git.pengutronix.de/git/barebox.git
git checkout -b work origin/next
```

2) Then you need to configure it

For the 64 MiB board

```
make usb_a9g20_defconfig
```

For the 128 MiB board

```
make usb_a9g20_128mib_defconfig
```

3) Compile it

```
make
```

4) Flash it

now you need to flash it via

SAM-BA

OpenOCD

5) start the board

```
barebox 2011.10.0-00144-gc9145e1 (Oct 25 2011 - 17:48:18)

Board: Calao USB-A9G20
Clocks: CPU 399 MHz, master 133 MHz, main 12.000 MHz
NAND device: Manufacturer ID: 0x2c, Chip ID: 0xaa (Micron NAND 256MiB 1,8V 8-bit)
Scanning device for bad blocks
Bad eraseblock 591 at 0x049e0000
Bad eraseblock 920 at 0x07300000
Bad eraseblock 1026 at 0x08040000
Bad eraseblock 1388 at 0x0ad80000
Bad eraseblock 1764 at 0x0dc80000
Malloc space: 0x23500000 -> 0x23f00000 (size 10 MB)
Stack space : 0x234f8000 -> 0x23500000 (size 32 kB)
envfs: wrong magic on /dev/env0
no valid environment found on /dev/env0. Using default environment
running /env/bin/init...

Hit any key to stop autoboot:  0
barebox@Calao USB-A9G20:/
#
```

Linux

You can use the mainline kernel (merge in release 3.1)

1) First you need to clone the tree

```
git clone git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux-2.6.git
git checkout -b work
```

2) Then you need to configure it

currently no defconfig is merge for the next release you could use at91sam9g20_defconfig but right now you need to use at91sam9g20ek_defconfig and select it via Kconfig "CALAO USB-A9G20"

```
make at91sam9g20ek_defconfig
make menuconfig
```

or

```
make at91sam9g20ek_defconfig
sed -i -e "s:# CONFIG_MACH_USB_A9G20 is not set:CONFIG_MACH_USB_A9G20=y:g" .config
```

3) Compile it

```
make uImage
```

4) Run it

You can run it via nfs or tftp from barebox

for this you need to modify the /env/config

a) on board

via

```
ed /env/config
```

b) on pc

by editing

```
diff --git a/arch/arm/boards/usb-a926x/env/config b/arch/arm/boards/usb-a926x/env/config
index d77f678..660cb3c 100644
--- a/arch/arm/boards/usb-a926x/env/config
+++ b/arch/arm/boards/usb-a926x/env/config
@@ -2,16 +2,16 @@

 # use 'dhcp' to do dhcp in barebox and in kernel
 # use 'none' if you want to skip kernel ip autoconfiguration
-ip=dhcp
+#ip=dh

 # or set your networking parameters here
-#eth0.ipaddr=a.b.c.d
-#eth0.netmask=a.b.c.d
-#eth0.gateway=a.b.c.d
-#eth0.serverip=a.b.c.d
+eth0.ipaddr=192.168.201.33
+eth0.netmask=255.255.255.0
+eth0.gateway=192.168.201.98
+eth0.serverip=192.168.201.98

 # can be either 'nfs', 'tftp', 'nor' or 'nand'
-kernel_loc=tftp
+kernel_loc=nfs
 # can be either 'net', 'nor', 'nand' or 'initrd'
 rootfs_loc=net

@@ -29,6 +29,9 @@ kernelimage=uImage
 #kernelimage_type=raw_lzo
 #kernelimage=Image.lzo

+nfsroot="${eth0.serverip}:/opt/work/buildroot/build/usb_9g20/target/"
+kernelimage="/opt/work/buildroot/build/usb_9g20/images/uImage-usb-a9g20"
+
 nand_device=atmel_nand
 nand_parts="128k(at91bootstrap),256k(barebox)ro,128k(bareboxenv),128k(bareboxenv2),4M(kernel),120M(rootfs),-(data)"
 rootfs_mtdblock_nand=5
```

5) start the board

Note that the kernel you see in this log is a development kernel using the Device tree

nb: sometime the nfs timeout to boot again just type boot enter

```
barebox 2011.10.0-00144-gc9145e1 (Oct 25 2011 - 17:48:18)

Board: Calao USB-A9G20
Clocks: CPU 399 MHz, master 133 MHz, main 12.000 MHz
NAND device: Manufacturer ID: 0x2c, Chip ID: 0xaa (Micron NAND 256MiB 1,8V 8-bit)
Scanning device for bad blocks
Bad eraseblock 591 at 0x049e0000
Bad eraseblock 920 at 0x07300000
Bad eraseblock 1026 at 0x08040000
Bad eraseblock 1388 at 0x0ad80000
Bad eraseblock 1764 at 0x0dc80000
Malloc space: 0x23500000 -> 0x23f00000 (size 10 MB)
Stack space : 0x234f8000 -> 0x23500000 (size 32 kB)
envfs: wrong magic on /dev/env0
no valid environment found on /dev/env0. Using default environment
running /env/bin/init...

Hit any key to stop autoboot:  0
warning: No MAC address set. Using random address D2:E7:9F:56:A2:96
phy0: Link is up - 100/Full
T T
Filename '/opt/work/buildroot/build/usb_9g20/images/uImage-usb-a9g20'.
        [#################################################################NFS failed: Interrupted system call
```

```
barebox@Calao USB-A9G20:/
# boot
Filename '/opt/work/buildroot/build/usb_9g20/images/uImage-usb-a9g20'.
       [#################################################################]
booting kernel of type uimage from /dev/ram0.kernel
   Verifying Checksum ... OK
   Image Name:   Linux-3.0.0+
   Created:      2011-10-24  21:55:00 UTC
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    1988582 Bytes =  1.9 MB
   Load Address: 20008000
   Entry Point:  20008000
OK

Starting kernel ...

commandline: console=ttyS0,115200 ip=192.168.201.33:192.168.201.98:192.168.201.98:255.255.255.0::eth0: root=/dev/nfs
nfsroot=192.168.201.98:/opt/work/buildroot/build/usb_9g20/target/,v3,tcp noinitrd mtdparts=atmel_nand:128k(at91bootst
rap),256k(barebox)ro,128k(bareboxenv),128k(bareboxenv2),4M(kernel),120M(rootfs),-(data)
arch_number: 1841
Uncompressing Linux... done, booting the kernel.
Linux version 3.0.0+ (root@j-debian) (gcc version 4.5.1 (Sourcery G++ Lite 2010.09-50) ) #66 Tue Oct 25 05:54:39 CST
2011
CPU: ARM926EJ-S [41069265] revision 5 (ARMv5TEJ), cr=00053177
CPU: VIVT data cache, VIVT instruction cache
Machine: Atmel AT91SAM (Device Tree), model: Calao USB A9G20
Memory policy: ECC disabled, Data cache writeback
AT91: Detected soc type: at91sam9g20
AT91: Detected soc subtype: Unknown
AT91: sram at 0x200000 of 0x4000 mapped at 0xfef74000
AT91: sram at 0x300000 of 0x4000 mapped at 0xfef70000
Clocks: CPU 399 MHz, master 133 MHz, main 12.000 MHz
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 16256
Kernel command line: console=ttyS0,115200 ip=192.168.201.33:192.168.201.98:192.168.201.98:255.255.255.0::eth0: root=/
dev/nfs nfsroot=192.168.201.98:/opt/work/buildroot/build/usb_9g20/target/,v3,tcp noinitrd mtdparts=atmel_nand:128k(at
91bootstrap),256k(barebox)ro,128k(bareboxenv),128k(bareboxenv2),4M(kernel),120M(rootfs),-(data)
PID hash table entries: 256 (order: -2, 1024 bytes)
Dentry cache hash table entries: 8192 (order: 3, 32768 bytes)
Inode-cache hash table entries: 4096 (order: 2, 16384 bytes)
Memory: 64MB = 64MB total
Memory: 60960k/60960k available, 4576k reserved, 0K highmem
Virtual kernel memory layout:
    vector  : 0xffff0000 - 0xffff1000   (   4 kB)
    fixmap  : 0xfff00000 - 0xfffe0000   ( 896 kB)
    DMA     : 0xffc00000 - 0xffe00000   (   2 MB)
    vmalloc : 0xc4800000 - 0xfee00000   ( 934 MB)
    lowmem  : 0xc0000000 - 0xc4000000   (  64 MB)
    modules : 0xbf000000 - 0xc0000000   (  16 MB)
      .text : 0xc0008000 - 0xc0388620   (3586 kB)
      .init : 0xc0389000 - 0xc03a7000   ( 120 kB)
      .data : 0xc03a8000 - 0xc03cc240   ( 145 kB)
       .bss : 0xc03cc264 - 0xc03dd8fc   (  70 kB)
NR_IRQS:192
looking for phys_base=fffff000, irq_start=0
AT91: 96 gpio irqs in 3 banks
Console: colour dummy device 80x30
Calibrating delay loop... 199.06 BogoMIPS (lpj=995328)
pid_max: default: 32768 minimum: 301
Mount-cache hash table entries: 512
CPU: Testing write buffer coherency: ok
pinctrl core: initialized pinctrl subsystem
NET: Registered protocol family 16
0x0
0xfffffff0
AT91: Power Management
AT91: Starting after general reset
 pinctrl.0: hogged map DGBU_RX, function dbgu_rxd
 pinctrl.0: hogged map DGBU_TX, function dbgu_txd
pinmux-at91 pinmux.2: initialized AT91 pinmux driver
atmel_tcb: probe of atmel_tcb.1 failed with error -22
bio: create slab <bio-0> at 0
SCSI subsystem initialized
```

```
usbcore: registered new interface driver usbfs
usbcore: registered new interface driver hub
usbcore: registered new device driver usb
Advanced Linux Sound Architecture Driver Version 1.0.24.
cfg80211: Calling CRDA to update world regulatory domain
Switching to clocksource tcb_clksrc
NET: Registered protocol family 2
IP route cache hash table entries: 1024 (order: 0, 4096 bytes)
TCP established hash table entries: 2048 (order: 2, 16384 bytes)
TCP bind hash table entries: 2048 (order: 1, 8192 bytes)
TCP: Hash tables configured (established 2048 bind 2048)
TCP reno registered
UDP hash table entries: 256 (order: 0, 4096 bytes)
UDP-Lite hash table entries: 256 (order: 0, 4096 bytes)
NET: Registered protocol family 1
RPC: Registered named UNIX socket transport module.
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
RPC: Registered tcp NFSv4.1 backchannel transport module.
NetWinder Floating Point Emulator V0.97 (double precision)
JFFS2 version 2.2. (NAND) (SUMMARY)  © 2001-2006 Red Hat, Inc.
msgmni has been set to 119
io scheduler noop registered (default)
fffff200.serial: ttyS0 at MMIO 0xfffff200 (irq = 1) is a ATMEL_SERIAL
console [ttyS0] enabled
brd: module loaded
loop: module loaded
No SmartMedia card inserted.
ohci_hcd: USB 1.1 'Open' Host Controller (OHCI) Driver
Initializing USB Mass Storage driver...
usbcore: registered new interface driver usb-storage
USB Mass Storage support registered.
mousedev: PS/2 mouse device common for all mice
ALSA device list:
  No soundcards found.
TCP cubic registered
NET: Registered protocol family 17
lib80211: common routines for IEEE802.11 drivers
/opt/work/linux-2.6/drivers/rtc/hctosys.c: unable to open rtc device (rtc0)
```

RootFS

You can use multiple tools to build the rootfs

```
* Debian
* OpenEmbedded
* BuildRoot
```

OpenOCD

On the Board you will find a FDTI FT2232D that it's use as a UART and the JTAG.

If you have a old kernel the ftdi drivers will probe the two port of the FT2232D as UART

(ttyUSB0 as JTAG and ttyUSB1 as UART)

So you will need to patch your kernel with http://lkml.org/lkml/2011/9/28/466

or use a out-of-tree module like this tree git://uboot.jcrosoft.org/ftdi-module.git

now you have a working environment you can use OpenOCD

Most of the recent distribution now use the last release of OpenOCD we did which is currently the v0.5.0

How to use it?

OpenOCD need two part the

1) the Interface configuration

This the description of your ICE/JTAG

```
cat calao-usb-a9g20-lpw.cfg
#
# CALAO Systems USB-A9G20-C02
#
# http://www.calao-systems.com/
#

interface ft2232
ft2232_device_desc "USB-A9G20-LPW"
ft2232_layout jtagkey
ft2232_vid_pid 0x0403 0x6010
```

2) the board configuration

```
cat usb-a9g20-lpw.cfg
#################################################################################################
#                                                                                               #
# Author: gregory Hermant (gregory.hermant@calao-systems.com)                                   #
#         Jean-Christophe PLAGNIOL-VILLARD <plagnioj@jcrsoft.com>                               #
# Generated for the USB-A9G20-LPW using Amontec JTAGKEY probe                                   #
#                                                                                               #
#################################################################################################

# We add to the minimal configuration.
source [find mem_helper.tcl]
source [find target/at91sam9g20.cfg]

reset_config trst_and_srst

# Use caution changing the delays listed below.  These seem to be affected by the board and type of
# debugger dongle.  A value of 200 ms seems to work reliably for the configuration listed in the file header above.

jtag_nsrst_delay 200
jtag_ntrst_delay 200

scan_chain
#$_TARGETNAME configure -event gdb-attach { reset init }
#$_TARGETNAME configure -event reset-start {at91sam9g20_reset_start}
#$_TARGETNAME configure -event reset-init {at91sam9g20_reset_init}

proc at91sam9g20_reset_start { } {

        # Make sure that the the jtag is running slow, since there are a number of different ways the board
        # can be configured coming into this state that can cause communication problems with the jtag
        # adapter.  Also since this call can be made following a "reset init" where fast memory accesses
        # are enabled, need to temporarily shut this down so that the RSTC_MR register can be written at slower
        # jtag speed without causing GDB keep alive problem.

        arm7_9 fast_memory_access disable
        jtag_khz 2                      ;# Slow-speed oscillator enabled at reset, so run jtag speed slow.
        halt 10000                              ;# Make sure processor is halted, or error will result in following s
teps.
        wait_halt 1000000
        mww 0xfffffd08 0xa5000501       ;# RSTC_MR : enable user reset.
}

proc at91sam9g20_reset_init { } {

        # At reset AT91SAM9G20 chip runs on slow clock (32.768 kHz).  To shift over to a normal clock requires
        # a number of steps that must be carefully performed.  The process outline below follows the
        # recommended procedure outlined in the AT91SAM9G20 technical manual.
        #
        # Several key and very important things to keep in mind:
        # The SDRAM parts used currently on the Atmel evaluation board are -75 grade parts.  This
        # means the master clock (MCLK) must be at or below 133 MHz or timing errors will occur.  The processor
        # core can operate up to 400 MHz and therefore PCLK must be at or below this to function properly.
```

```
        mww 0xfffffd44 0x00008000       # WDT_MR : disable watchdog.

        # Set oscillator bypass bit (12.00 MHz external oscillator) in CKGR_MOR register.

        mww 0xfffffc20 0x00000002

        # Set PLLA Register for 798.000 MHz (divider: bypass, multiplier: 132).
        # Wait for LOCKA signal in PMC_SR to assert indicating PLLA is stable.

        mww 0xfffffc28 0x20843F02
        while { [expr [mrw 0xfffffc68] & 0x02] != 2 } { sleep 1 }

        # Set master system clock prescaler divide by 6 and processor clock divide by 2 in PMC_MCKR.
        # Wait for MCKRDY signal from PMC_SR to assert.

        mww 0xfffffc30 0x00001300
        while { [expr [mrw 0xfffffc68] & 0x08] != 8 } { sleep 1 }

        # Now change PMC_MCKR register to select PLLA.
        # Wait for MCKRDY signal from PMC_SR to assert.

        mww 0xfffffc30 0x00001302
        while { [expr [mrw 0xfffffc68] & 0x08] != 8 } { sleep 1 }

        # Processor and master clocks are now operating and stable at maximum frequency possible:
        #       -> MCLK = 133.000 MHz
        #       -> PCLK = 400.000 MHz

        # Switch over to adaptive clocking.

        jtag_khz 0

        # Enable faster DCC downloads.

        arm7_9 dcc_downloads enable

        # To be able to use external SDRAM, several peripheral configuration registers must
        # be modified.  The first change is made to PIO_ASR to select peripheral functions
        # for D15 through D31.  The second change is made to the PIO_PDR register to disable
        # this for D15 through D31.

        mww 0xfffff870 0xffff0000
        mww 0xfffff804 0xffff0000

        # The EBI chip select register EBI_CS must be specifically configured to enable the internal SDRAM controller
        # using CS1.  Additionally we want CS3 assigned to NandFlash.  Also VDDIO is connected physically on
        # the board to the 1.8V VDC power supply so set the appropriate register bit to notify the micrcontroller.

        mww 0xffffef1c 0x000000a

        # The USB-A9G20-LPW Embedded computer has built-in NandFlash.  The exact physical timing characteristics
        # for the memory type used on the current board (MT29F2G08AACWP) can be established by setting
        # four registers in order:  SMC_SETUP3, SMC_PULSE3, SMC_CYCLE3, and SMC_MODE3.

        mww 0xffffec30 0x00020002
        mww 0xffffec34 0x04040404
        mww 0xffffec38 0x00070007
        mww 0xffffec3c 0x00030003

        # Now setup SDRAM.  This is tricky and configuration is very important for reliability!  The current calculat
ions
        # are based on 2 x Micron LPSDRAM MT48H16M16LFBF-75 memory (4 M x 16 bit x 4 banks).  If you use this file as
 a reference
        # for a new board that uses different SDRAM devices or clock rates, you need to recalculate the value inserte
d
        # into the SDRAM_CR register.  Using the memory datasheet for the -75 grade part and assuming a master clock
        # of 133.000 MHz then the SDCLK period is equal to 7.6 ns.  This means the device requires:
        #
        #       CAS latency = 3 cycles
        #       TXSR = 10 cycles
        #       TRAS = 6 cycles
```

```
        #       TRCD = 3 cycles
        #       TRP = 3 cycles
        #       TRC = 9 cycles
        #       TWR = 2 cycles
        #       9 column, 13 row, 4 banks
        #       refresh equal to or less then 7.8 us for commerical/industrial rated devices
        #
        #       Thus SDRAM_CR = 0xa6339279

        mww 0xffffea08 0xa6339279

        # Memory Device Type: Low-power SDRAM
        mww 0xffffea24 0x00000001

        # Next issue a 'NOP' command through the SDRAMC_MR register followed by writing a zero value into
        # the starting memory location for the SDRAM.

        mww 0xffffea00 0x00000001
        mww 0x20000000 0

        # Issue an 'All Banks Precharge' command through the SDRAMC_MR register followed by writing a zero
        # value into the starting memory location for the SDRAM.

        mww 0xffffea00 0x00000002
        mww 0x20000000 0

        # Now issue an 'Auto-Refresh' command through the SDRAMC_MR register.  Follow this operation by writing
        # zero values eight times into the starting memory location for the SDRAM.

        mww 0xffffea00 0x4
        mww 0x20000000 0
        mww 0x20000000 0
        mww 0x20000000 0
        mww 0x20000000 0
        mww 0x20000000 0
        mww 0x20000000 0
        mww 0x20000000 0
        mww 0x20000000 0

        # Almost done, so next issue a 'Load Mode Register' command followed by a zero value write to the
        # the starting memory location for the SDRAM.

        mww 0xffffea00 0x3
        mww 0x20000000 0

        # Signal normal mode using the SDRAMC_MR register and follow with a zero value write the the starting
        # memory location for the SDRAM.

        mww 0xffffea00 0x0
        mww 0x20000000 0

        # Finally set the refresh rate to about every 7 us (7.5 ns x 924 cycles).

        mww 0xffffea04 0x0000039c
}
```

3) start OpenOCD

now you need so start OpenOCD. For this you need to specify the interface (First) and then the board

```
openocd -f tcl/interface/calao-usb-a9g20-lpw.cfg  -f tcl/board/usb-a9g20-lpw.cfg
Open On-Chip Debugger 0.5.0 (2011-08-09-08:45)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.berlios.de/doc/doxygen/bugs.html
Info : only one transport option; autoselect 'jtag'
trst_and_srst separate srst_gates_jtag trst_push_pull srst_open_drain
adapter_nsrst_delay: 300
jtag_ntrst_delay: 200
RCLK - adaptive
RCLK - adaptive
trst_and_srst separate srst_gates_jtag trst_push_pull srst_open_drain
adapter_nsrst_delay: 200
jtag_ntrst_delay: 200
   TapName             Enabled  IdCode     Expected   IrLen IrCap IrMask
-- ------------------- -------- ---------- ---------- ----- ----- ------
 0 at91sam9g20.cpu        Y     0x00000000 0x0792603f     4 0x01  0x0f
at91sam9g20_init
Info : max TCK change to: 30000 kHz
Info : RCLK (adaptive clock speed)
Info : JTAG tap: at91sam9g20.cpu tap/device found: 0x0792603f (mfg: 0x01f, part: 0x7926, ver: 0x0)
Info : Embedded ICE version 6
Info : at91sam9g20.cpu: hardware has 2 breakpoint/watchpoint units
```

you can now connect to you board via gdb or telnet

a) telnet

```
telnet 127.0.0.1 4444
Trying 127.0.0.1...
Connected to 127.0.0.1.
Escape character is '^]'.
Open On-Chip Debugger
> halt
target state: halted
target halted in ARM state due to debug-request, current mode: Supervisor
cpsr: 0x600000d3 pc: 0x23f0457c
MMU: disabled, D-Cache: disabled, I-Cache: enabled
>
```

b) gdb

```
$ arm-none-linux-gnueabi-gdb
GNU gdb (Sourcery G++ Lite 2010.09-50) 7.2.50.20100908-cvs
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-pc-linux-gnu --target=arm-none-linux-gnueabi".
For bug reporting instructions, please see:
<https://support.codesourcery.com/GNUToolchain/>.
(gdb) target remote :3333
Remote debugging using :3333
0x23f07f74 in ?? ()
```

note that the current board script does not init the hardware when you connect to gdb

for this you will need to use the monitor

```
monitor reset init halt
```

sometime the halt timeout in this case just do

```
monitor halt
```

Category:

- ARM Development Boards

From: eLinux.org

CR48

Contents

Project Summary



The Cr-48 is the test notebook Google designed for the Pilot program. It's the first of its kind.

It's ready when you are, booting in about 10 seconds and resuming from sleep instantly. There's built-in Wi-Fi and 3G, so you can stay connected everywhere, and a webcam for video chat. The vibrant 12-inch LCD display, full-size keyboard and oversized touchpad let you enjoy the web comfortably. And at just 3.8 pounds with over eight hours of active usage and a week of standby time, it's easy to take along for the ride.

What did we leave out? Spinning disks, caps-lock key, function keys, and lap burns.

The Cr-48 is available exclusively to participants in the Pilot program.

The pilot program is available at: http://www.google.com/chromeos/pilot-program.html

The CR48 runs Google's Chrome Operating system. None of your files are stored on the PC, they're all stored in the cloud.

To keep it synced with the cloud, it includes a wifi connection, and a verizon 3G modem (which comes with 100MB/month for 2 years, for beta testers).

Hardware

- Processor: Intel Atom Processor N455 1.66GHz 512K Cache
- Chipset: Intel CG82NM10 PCH
- Motherboard: Tripod Motherboard MARIO – 6050A240910 – MB – A03
- Ram: Hynix 2GB DDR3 1Rx8 PC3 – 10600S Ram
- Read Only Memory: ITE IT8500E Flash ROM
- SSD Drive: SanDisk sdsa4dh-016G 16GB SATA SSD
- Wireless Wan: Qualcomm Gobi2000 PCI Express Mini Card
- 3g Adapter: AzureWave 802.11 a/b/g/n PCI-E Half MiniCard
- Bluetooth: Atheros AR5BBU12 Bluetooth V2.1 EDR
- GPS: Gobi 2000 3G Modem
- Integrated Webcam: VGA - 640x480
- Screen
  - Display: 12.1 inches
  - Screen Resolution: 1,280X800 pixels
- Keyboard: 74 keys
- Pointing device: Track-pad (Multi-touch)
- Ambient light sensor

Ports

- 1 USB (right side)
- 1 SD (right side)
- 1 3.5mm headphone jack (right side)
- 1 Power jack (right side)
- 1 VGA (left side)
- 1 Sim card slot (underneath the battery)

Software Revisions

Beta Channel

- 0.9.128.12
- 0.9.128.14

Dev Channel

- 0.10.146.1
- 0.10.156.1
- 0.10.156.4
- 0.10.156.18
- 0.10.156.20

How to

Print

The CR-48 has no built in facilities for hooking directly to a printer, instead, it prints via Google Cloud print. Currently with cloud printing, you need to have a computer hooked to your printer that is currently powered on.

- Setting up printing with a Windows based computer
- Setting up printing with a Linux based computer

Access Hardware

- Bluetooth
- GPS
- SIM Slot

Install Different Programs

- Locate
- Nano text editor
- VNC viewer

Root the Device

- Root the CR48
- Make the /mnt/stateful partition exec friendly

Tips and Tricks

- About: Pages
- Copy and paste (also paste to the terminal)
- Force a Software Update

Common Problems

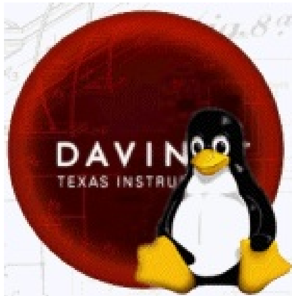
- Static in audio after update (SOLVED in latest updates)
- Timezone wont change (SOLVED in latest updates)
- Google searches crash (SOLVED in latest updates)
- Proxy settings take forever to detect (SOLVED in latest updates)

Category:

- CR48

> From: eLinux.org

DaVinci



Texas Instruments DaVinci™ is a family of multimedia system on chip processors for embedded application. Most of DaVinci processors based of of ARM and DSP microprocessor cores.

DaVinci SDK uses U-Boot, MontaVista Linux, Open_Embedded

Contents

Important resources

- DaVinci™ Digital Media Processors
- http://wiki.davincidsp.com/
- Forums
- http://feeds.feedburner.com/tidavincitechnology
    - news feed
- http://www.linux-davinci.info/
- Old DaVinci Pages Obsolete and/or Out of Date

Programming



- Blinking LEDS on the EVM over I2c - EVM_LED_Blinking.c



- Simple TTY host program to access serial port - EVM_comz.c



- PWM kernel driver (char driver) - PWM.c *Good example for your own device drivers*



- Accessing an I2C temperature sensor - temp_sens.c



- Useful I2C routines - I2Croutines.c

 - Overclocking the EVM EVMoc

Hardware Hacks

 - Adding a second MMC / SD slot - Second MMC / SD

 - Installing some input buttons - Input buttonz

 - Adding I2C devices (e.g. Temp Sensor) - I2C Mods

Mailing Lists and Useful Links

- http://linux.davincidsp.com/mailman/listinfo/davinci-linux-open-source or the Searchable archive
- http://source.mvista.com/git/ - The montavista GIT repository
- http://www.applieddata.net/forums/topic.asp?TOPIC_ID=2024
    - U-Boot configuration scripts for flash filesystems

Misc Info

- DaVinci_Initrd_1.0
- DaVinci_U-boot_1.0
- DaVinci_USBHost_1.0

Categories:

- OMAP
- DaVinci

> From: eLinux.org

Devkit8000

Contents

Devkit8000 Board Overview

Embest DevKit8000 Evaluation Board is a compact board using TI's OMAP3530 600MHz ARM Cortex-A8 (600MHz ARM Cortex-A8 core paired with a 430MHz TMS320C64x+ DSP core) microprocessor. It takes full features of this processor and supports up to 256MByte DDR SDRAM and 256MByte NAND Flash as well as high-speed USB2.0 OTG function. The board has exposed many other hardware interfaces including RS232 serial port, LCD/TSP, DVI-D, S-Video, Ethernet, SD, keyboard, camera, SPI, I2C and JTAG. The board has two methods to boot the system from either SD card or NAND flash. It is able to support WinCE and Linux OS and provided with WinCE6.0 BSP and Linux2.6.28 BSP. Embest also provides demos of Google Android OS, Angstrom (GPE) and DVSDK for user experience.

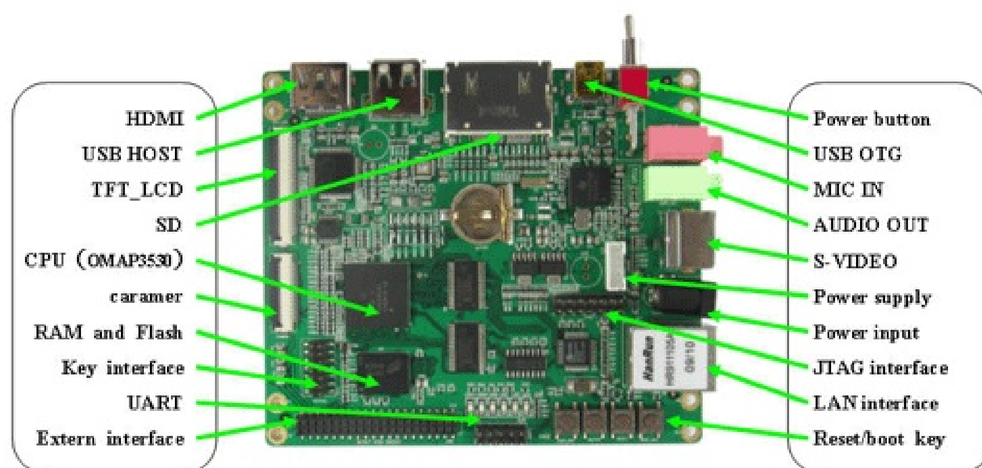


Devkit8000 Evaluation Board

Devkit8000 + 4.3 inch LCD Android Demo Display

Hardware

Onboard Interfaces and Connectors



Hardware Features

- Dimensions: 110mm x 95mm
- Working temperature: 0°C to 70°C
- Processor: TI OMAP3530 microprocessor with 600MHz ARM Cortex-A8 RISC Core
- Power supply: +5V
- 256MB DDR SDRAM, 166MHz
- 256MB NAND Flash, 16bit
- LCD/Touch Screen interface (50-pin FPC connector, support resolution up to 2048*2048)
- DVI high-resolution image output port (HDMI interface, support 720p, 30fps signal)
- S-Video display interface
- One audio input interface (3.5mm audio jack)
- One 2-channel audio output interface (3.5mm audio jack)
- One 10/100M Ethernet interface (RJ45)
- One High-speed USB2.0 OTG port (Mini USB type interface)
- One High-speed USB2.0 Host port (USB A type interface)
- Two serial ports (one 3-wire RS232 serial port led out from 2.54mm 10-pin connector and one 5-wire TTL serial port led out from expansion connector)
- SD card interface (supports 3.3V and 1.8V logic voltage)
- One camera interface (30-pin FPC connector, support CCD or CMOS camera)
- 6*6 keyboard interface
- One 14-pin Jtag interface
- Four buttons (Reset, Boot, User defined, On/Off)
- One expansion connector (2.0mm 40-pin SMT Female Pin Header, McSPI, McBSP, I2C, HDQ, GPIO are led out from this connector)

Main Chip Introduction

DevKit8000	Chip	Remark
Processor	OMAP3530CUS	0.65mm CUS package
Memory	MT29C2G48MAKLCJA-6IT	256MB DDR/256MB NAND Flash, 137-Ball TFBGA, Mark:JW305, Micron
Power Management Chip	TPS65930BZCH	Extended power management, RTC, USB OTG, Audio, 6*6 Keyboard
DVI-D	TFP410	HDMI connector, output DVI-D signal, not including audio
Ethernet	DM9000	RJ45, 10M/100M adaptive, at present testing speed can reach 36M
Serial port	MAX3232 CSE	RS232

Video Display

DVI-D

DVI high-resolution display (HDMI interface, 30fps signal), Embest provides a HDMI to DVI-D cable in complete configuration for connection. Signal does not include audio signal.

S-video

S-video display, can output video signal but not including audio signal.

LCD

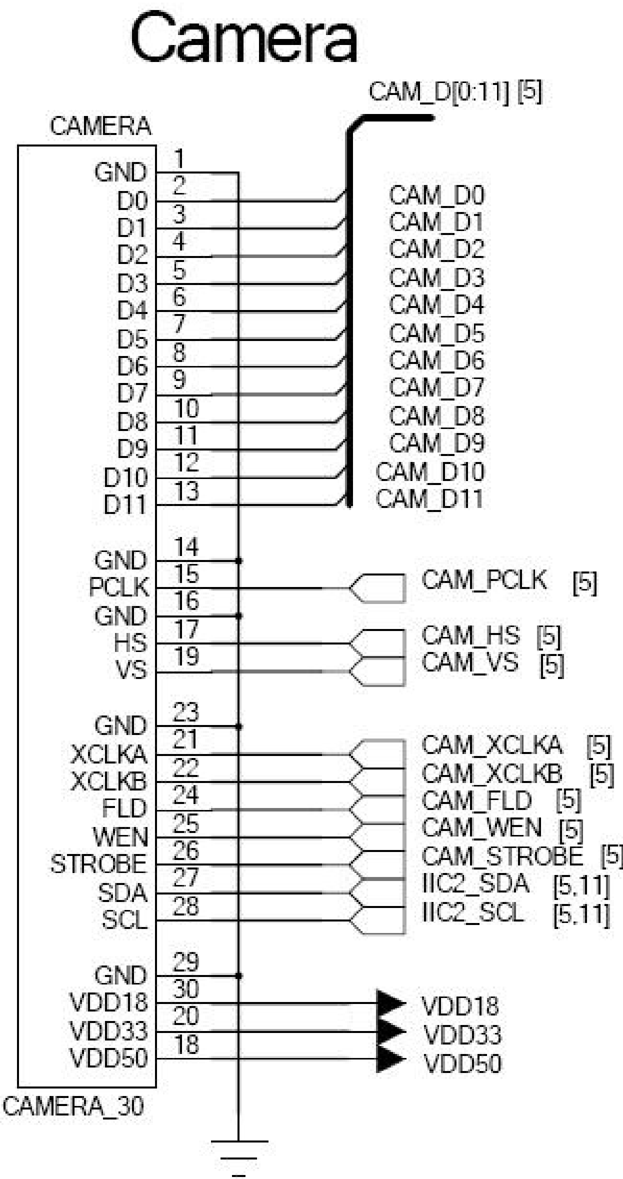
24-bit LCD display, can output true-color RGB signal, R:B:G=8:8:8, can support resolution up to 2048*2048 pixels. (50-pin 0.5mm pitch FPC connector. Interface Singal Type

RGB data signal	LCD control signal	SPI signal	IIC signal	Touch Screen signal	Voltage output
24bit	6bit	4bit	2bit	4bit	5bit
R:G:B=8:8:8	row control signal	standard spi signal	2-bit IIC signal	4-wire touch screen	output voltage 5V, 3.3V, 1.8V

USB OTG (Mini-AB) Wire Map

1: VB 2: D- 3: D+ 4: ID 5: GND User needs only to short circuited 4 and 5.

Camera Interface



Expansion Interface

Embest Devkit8000 uses a 2.0mm 40-pin SMT Female Pin Header to bring out McSPI, McBSP, I2C, HDQ, GPIO from this interface

PIN	Signal	Function Description
1	GND	GND
2	BSP1_DX	Transmitted serial data 1
3	BSP1_DR	Received serial data 1
4	BSP1_CLKR	Received clock 1
5	BSP1_FSX	Transmit frame synchronization 1
6	BSP1_CLKX	Transmit clock 1
7	BSP1_CLKS	External clock input 1
8	BSP1_FSR	Receive frame synchronization 1
9	UART1_CTS	UART1 clear to send

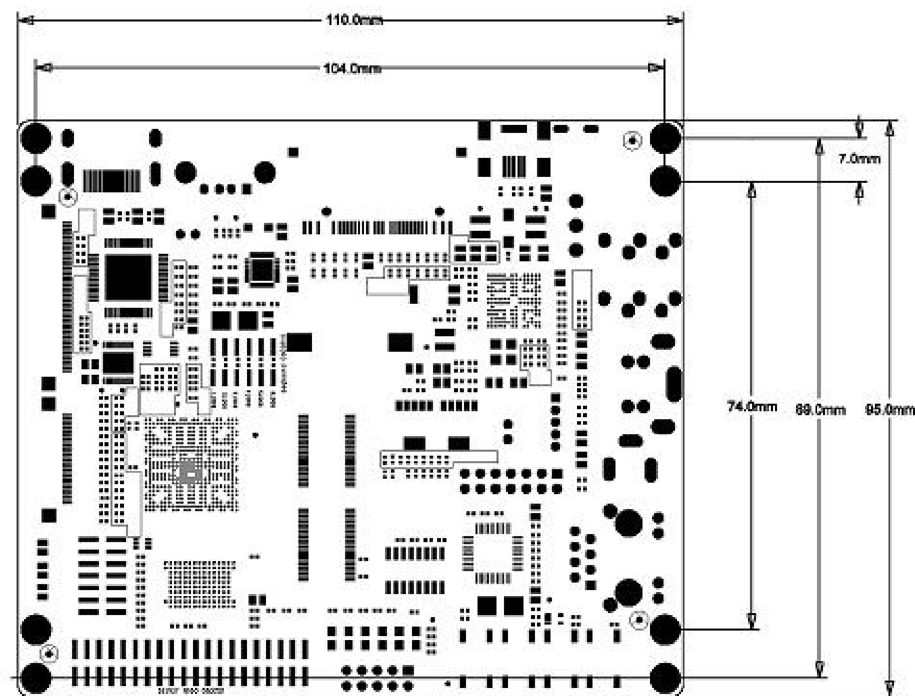
10	UART1_RTS	UART1 request to send
11	UART1_RX	UART1 receive data
12	UART1_TX	UART1 transmit data
13	GND	GND
14	MMC2_CLK	MMC2 card clock
15	MMC2_CMD	MMC2 card command
16	MMC2_D0	MMC2 card data 0
17	MMC2_D1	MMC2 card data 1
18	MMC2_D2	MMC2 card data 2
19	MMC2_D3	MMC2 card data 3
20	MMC2_D4	MMC2 card data 4
21	MMC2_D5	MMC2 card data 5
22	MMC2_D6	MMC2 card data 6
23	MMC2_D7	MMC2 card data 7
24	BSP3_DX	Transmitted serial data 3
25	BSP3_DR	Received serial data 3
26	BSP3_CLKX	Transmit clock 3
27	BSP3_FSX	Transmit frame synchronization 3
28	GND	GND
29	IIC3_SCL	IIC3 master serial clock
30	IIC3_SDA	IIC3 serial bidirectional data
31	SPI1_SIMO	Slave data in, master data out
32	SPI1_SOMI	Slave data out, master data in
33	SPI1_CLK	SPI1 clock
34	SPI1_CS0	SPI enable 0
35	SPI1_CS3	SPI enable 3
36	HDQ_SIO	Bidirectional HDQ
37	VDD33	3.3V
38	VDD18	1.8V
39	VDD50	5V
40	VDD50	5V

LayOut

The Devkit8000 PCB has 6-layer design, each layer is layout as below:

1	2	3	4	5	6
Top	Ground	Singal	Power	Ground	Bottom

Dimension Drawing



Function Block Diagram

Optional Function Modules

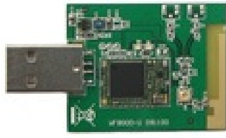
- **VGA Module** (VGA8000)


The VGA8000 module employs PHILIPS 74alvc164245 chip with 240MHz maximum sample rate. It can output standard LCD signal and display smoothly under 1024*768 resolution.

- **Analog Camera Module** (CAM8000-A)


The CAM8000-A module is designed for using on Devkit8000 Evaluation Board with standard 720*576 PAL resolution. It supports analog camera with BNC connector and connects to Devkit8000 board through an 30-pin FFC cable.

- **USB WiFi Module** (WF8000-U)

 The WF8000-U is a USB based WiFi module relying on the WiFi IEEE 802.11 standards. It is applied to be used on all Embest products which shall have USB interface. This module is hgihly integrated with MAC/BBP and 2.4GHz RF chip and builds a wireless network between PCs or some other terminals.

- **GPS Module** (GPS8000-S)

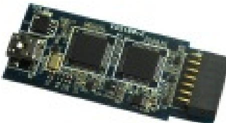
 The GPS8000-S is an ARM based embedded GPS module which allows to connect to Devkit8000 board to get position and altitude. It is a high sensitivity module with low power consumption. Devkit8000's UART (the serial port) from the expansion connector is used to communicate with the GPS through a tiny patch board.

- **GPRS Module** (GPRS8000-S)

 The GPRS8000-S is an ARM based embedded GPS module which allows to connect to Embest Devkit8000 board for GSM/GPRS solution. Devkit8000's UART (the serial port) from the expansion connector is used to communicate with the GPRS module.

- **CDMA8000-U Module** (CDMA8000-U)
- **WCDMA8000-U Module** (WCDMA8000-U)
- **Digital Camera Module** (CAM8100-U)

JTAG Tool



- XDS100v2 - the second release of the XDS100 JTAG emulator technology supporting debug of a variety of TI devices.

Available from Embest

Software

Software Features

Linux and WinCE OS support

Embest DevKit8000 supports for both Linux2.6.28 and WinCE6.0 operating systems.

OS	Item	Features	Description
Linux	Boot	Version	x-load-1.41, u-boot 1.3.3
		Boot Mode	Boot Linux from SD card, NAND Flash or Ethernet
		Image Update	Support updating image from SD card or Ethernet
	Kernel and Drivers	Version	Linux 2.6.28
		Support file systems	ROM/CRAM/EXT2/EXT3/FAT/NFS/JFFS2/UBIFS
		Drivers	Serial, RTC, NET, Nand, LCD, Touch Screen, SD, USB Host, USB OTG, DVI, Keypad, LED, Audio
	File system	Format	Ramdisk, UBI
		Characteristic	Provided Lib (ALSA -lib, tslib, glibc), udev support
	Demo	Angstrom	Audio (XMMS), network (firefox), graphics editor (gimp) and document processing software (Abiword)
		Android	Google developed a platform based on Linux open-source mobile phone operating system.
		DVSDK	Support MPEG4, MPEG2, H264, mp3, aac audio/video formats and Codecs
WinCE	Boot	Version	x-load-1.41, Eboot
		Boot Mode	Boot WinCE from SD card, NAND Flash or Ethernet
		Image Update	Support updating image from SD card or Ethernet
	System	Characteristic	KITL kernel debug, Reboot, Watchdog, RTC
		Drivers	Display driver（DVI, TFT LCD）
			SD card, Keyboard, McSPI, McBSP, Audio, NET, NLED, USB Host, USB OTG, WiFi, GPS, GPRS, CDMA
		Function	Power Management (backlight drive, battery-driven, sleep/wake-up function)
			Hive registry support
			ROM file system support
		Software features	Mediaplayer 9.0, Word and Internet Explorer 6.0
			.NET Compact Framework 3.5

Now Linux2.6.29 is ported. Source code download from here

Ubuntu on Devkit8000 (for reference)

Please refer to Devkit8000 Ubuntu.

Port QT on Devkit8000 (for reference)

Please refer to Devkit8000 QT on how to port QT on Devkit8000.

Demo (Android/Angstrom/DVSDK)

Android



The DevKit8000 can support Android which is a software platform and operating system for mobile devices, based on the Linux kernel, developed by Google and later the Open Handset Alliance. The board can run various applications based on Android. It supports 4.3" and 7" TFT LCD display and touch panel function. It can use the built-in audio player of Android to play kinds of audio files and transfer data through SD card or USB OTG. Please refer to Devkit8000 Android on how to use and port.

Angstrom



DevKit8000 can display Angstrom system on a DVI_D monitor running 720p videos at 30fps. DevKit8000 can use various software of Angsrtom to implement file editing, internet surfing, audio and video files playing and graphics editing and more other functions. User can use USB mouser to operate on Angstrom system. Please refer to Devkit8000 Angstrom on how to use and port.

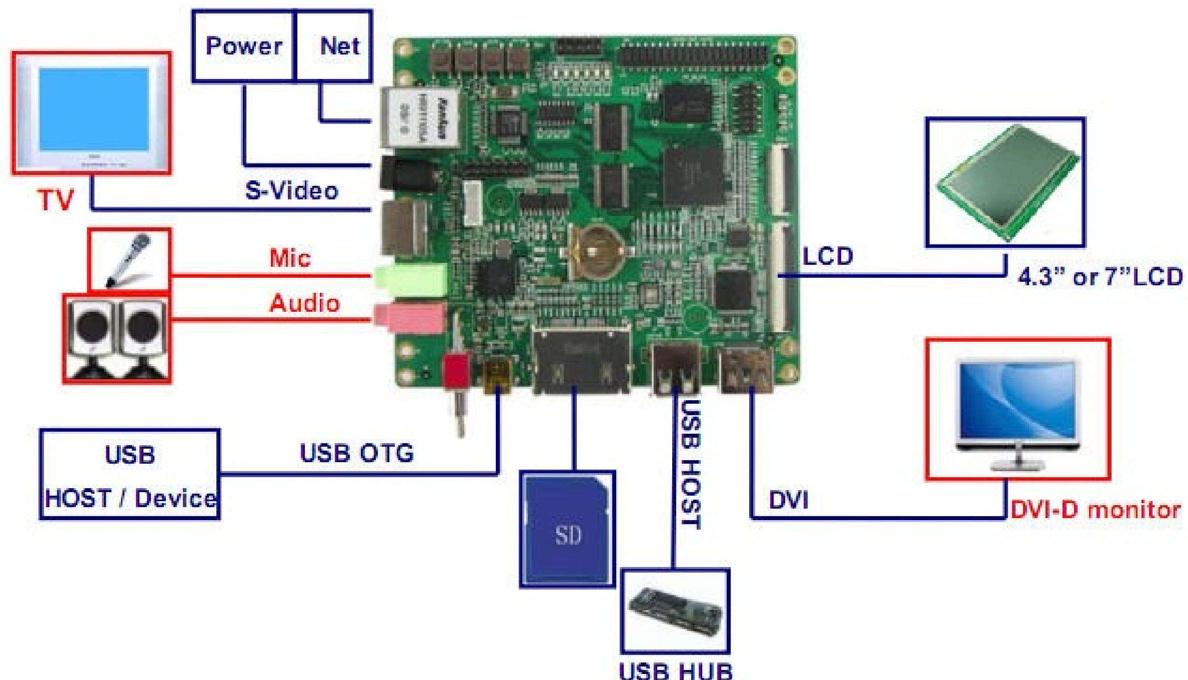
DVSDK (DSP)



DevKit8000 supports DVSDK package which includes following functions: Supports 2D/3D graphics acceleration; Supports DSP codec (Supports audio and video hardware decoding); Supports S-Video output. Please refer to Devkit8000_DVSDK on how to use and port.

Devkit8000 Evaluation Kit Overview

Embest DevKit8000 Evaluation Kit includes the DevKit8000 evaluation board and all necessary accessories to help users start their design of multimedia applications. The board is preloaded with Linux OS in NAND flash and WinCE OS in SD card. User can display the subsystem using a 4.3" TFT LCD and Touch screen or using a DVI-D monitor through an HDMI to DVI-D cable, or using TV for NTSC/PAL video out. The USB OTG interface can also be used as USB Host function with a Mini-A to Standard-A cable, or used as USB Device function with a Mini-B to Standard-A cable. Along with the kit, Embest provides user manual, schematic drawing and datasheet documents to help customers better understand and use the kit.



Configurations

Embest DevKit8000 Evaluation Kit has two configurations:

Standard Configuraiton

The standard configuration is focused on evaluating the basic functions of Devkit8000 board.



DevKit8000 Evaluation board



CD



Serial cable



5V@2A Power adapter



SD card

Complete Configuration

The complete configuration includes complete accessories for Devkit8000 which is convenient for special application development.



Cross Ethernet cable

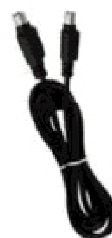


USB cable

A to Mini-A, A to Mini-B



Serial cable



S-video cable



HDMI to DVI-D cable



DevKit8000 Evaluation board



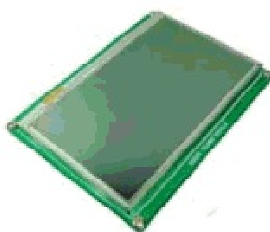
5V@2A Power adapter



USB HUB



SD card



4.3" or 7"LCD+Touch Screen



Touch Pen



CD

Product CD

The Devkit8000 product CD contains following contents:

- Linux2.6.28 source code (including driver, kernel, startup code and file system)
- WinCE6.0 BSP (including driver, kernel, startup code and file system all in source code)
- Schematic in PDF format
- Onboard chip datasheet
- Linux cross-compile toolchains
- User manual

Optional accessories

- CAM8000-A analog camera module
- Analog camera
- VGA8000 VGA module
- WF8000-U USB WiFi module
- CAM8100-U digital camera module
- GPS8000-S GPS module
- GPRS8000-S GPRS module
- CDMA8000-U USB 3G module (CDMA2000 standard)
- WCDMA8000-U USB 3G module (WCDMA standard)
- XDS100v2 USB JTAG emulator

FAQ

If you have got some problems when using Devkit8000, please refer to Devkit8000 FAQ.

Other Embest Products based on TI Processors

Embest has designed several products based on TI's ARM9 and ARM Cortex-A8 processors. Embestalso offers customer design service according to customer's requirements. Whether you need to reduce, add or modify to existing hardwares or generate a new solution, Embest will help customers with expert competence and rich experience.

- DevKit8500 Evalution Board based on TI's DM3730 or AM3715 ARM Cortex-A8 processor
- SBC8530 Single Board Computer based on TI's DM3730 ARM Cortex-A8 processor
- SBC8100 Plus Single Board Computer based on TI's DM3730 ARM Cortex-A8 processor
- SBC8018 Single Board Computer based on TI's AM1808 ARM926EJ-S processor
- SBC8100 Single Board Computer based on TI's OMAP3530 ARM Cortex-A8 processor
- SOC8200 Single Board Computer based on TI's AM3517 Sitara ARM Cortex-A8 processor
- Mini8510 Processor Card based on TI's DM3730 ARM Cortex-A8 processor
- Mini8100 Processor Card based on TI's OMAP3530 ARM Cortex-A8 processor

Categories:

- Development Boards
- OMAP
- Linux
- Devkit8000
- DevKit8500

> From: eLinux.org

Dragonboard

The **Dragonboard** is Qualcomm's publicly available development platform. As of spring 2014, there are four versions available based on three different systems-on-chip. The boards come pre-loaded with Android, but thanks to recent upstreaming work, it is increasingly possible to run the upstream Linux kernel on them.

Board	SOC	CPU	Pre-Installed OS	Upstream Device Tree
APQ8094	Snapdragon 810 (Cortex-A53/A57)	Octo-Core Cortex-A53/A57 big.LITTLE	Android 5.0	
APQ8074	Snapdragon 800 (Krait)	Quad-core Krait, 2.3 GHz	Android 4.2	qcom-apq8074-dragonboard.dtb
IFC6410	Snapdragon 600 (Krait)	Quad-core Krait, 1.7 GHz		qcom-apq8064-ifc6410.dtb
SYS6440	Snapdragon 600 (Krait)	Quad-core Krait, 1.7 GHz		
APQ8060A	Snapdragon S4 (Krait)	Dual-core Krait, 1.5 GHz		qcom-msm8660-surf.dtb
APQ8060	Snapdragon S3 (Scorpion)	Scorpion	Android 2.3	qcom-msm8660-surf.dtb

References

- http://www.mydragonboard.org/dragonboard-comparison/
- https://developer.qualcomm.com/sites/default/files/dragonboard-datasheet.pdf
- http://www.df.lth.se/~triad/krad/dragonboard/
- http://www.labradoc.com/i/follower/p/notes-qualcomm-apq8060-dragonboard
- https://www.codeaurora.org/contribute/projects/snapdragon-developer-platforms/

Category:

- Dragonboard

> From: eLinux.org

Embedded Open Modular Architecture/EOMA-68

\< Embedded Open Modular Architecture



Contents

EOMA-68 Introduction

The purpose of the EOMA68 specification is to bring simple robust mass-produced CPU Cards to end-users. To make end-users lives easier, purchasing decision-making should be made not on technical interface capabilities, neither should they be expected to have significant technological expertise. This is the primary reason why EOMA specifications have no optional interfaces of any kind. For this reason, for the lifetime of the specification (anticipated to be at least a decade) *all* CPU Cards compliant with the EOMA68 specification will be compatible with *all* compliant "Chassis" *(Note: "chassis" is the term used to describe a product into which a CPU Card is plugged).*

The trade-off between choosing a single-board design or even another modular form-factor is as follows:

- EOMA-68 products will not use all of the integrated functions of single-board designs, automatically resulting in minor cost increases for products.
- However, single-board designs are typically throw-away products where the lifetime of the product is critically dependent on an extremely fast-changing market.
- Single-board designs are typically throw-away hermetically-sealed products with neither user-serviceable nor user-replaceable parts
- EOMA-68 products are user-upgradeable. The Chassis can be kept out of landfill - kept in useful service - for the lifetime of its components. Only the CPU Card need be upgraded at a much lower cost than a single-board design in order to continuously give the product a new lease of life.
- EOMA-68 CPU Cards can be shared by the same end-user across multiple products, automatically resulting in a cost saving that far outweighs the minor overhead of a single EOMA68 system when compared to a single hermetically-sealed throw-away product.
- Old Chassis and CPU Cards can be re-purposed instead of discarded as e-waste.
- Q-Seven and other similar standards are not realistically user-upgradeable (not for the average person) because products require tools, technical knowledge in the selection of replacement parts, and expert knowledge in the handling of electronics before opening the case.
- EOMA68 products are upgraded by pushing a button and popping out the module: it literally takes seconds to install a new CPU Card.

So the benefits for end-users are very clear: EOMA-68 is easily understandable long-term investment for its end-users with significant long-term cost savings and reductions in e-waste. The benefits for factories are very clear: CPU Cards aggregated across multiple products means much better bulk purchasing power, and Chassis can also continue to be produced *without* requiring redesigns pretty much until the components go end-of-life.

No other modular computing standard available today has been designed with these aims in mind.

EOMA-68 Specification

This page describes the specification of EOMA-68. The number of pins on the interface is 68; the physical form-factor is the legacy PCMCIA.

Re-purposing of the PCMCIA interface and form-factor has been chosen to create portable mass-volume (100 million units and above) Embedded Computing Modules (Computer on Module). Mass-volume "Lowest Common Denominator" interfaces have been chosen, all of which have existed for over a decade, but are low-power enough to be standard across virtually all mass-produced powerful Embedded CPUs.

The interfaces are:

- 24-pin RGB/TTL (for LCD Panels and DVI/VGA/HDMI or other display conversion ICs)
- I2C
- 1st USB (Low Speed, Full Speed, optionally Hi Speed/480 Mbit/s and optionally USB3)
- 2nd USB (Low Speed, Full Speed, optionally Hi Speed/480 Mbit/s)
- 10/100 Ethernet (optionally 1,000 ethernet)
- 8 pins of General-purpose Digital I/O (GPIO) with multiplexing to SD/MMC and SPI on 6 pins

- 1 pin "External Interrupt" capable GPIO that will generate a fast hardware interrupt to the SoC
- SD/MMC (and down-level compatibility to SPI) multiplexed with 6 of the GPIO pins.
- TTL-compatible UART (Tx and Rx only).

These interfaces are **NOT OPTIONAL** for CPU Cards. All CPU Cards **MUST** provide **all** interfaces. I/O Boards on the other hand are free to implement whichever interfaces are required for the device. For example: whilst all CPU Cards **must** have an Ethernet interface, devices such as tablets or laptops into which CPU Cards are plugged are not *required* to have an ethernet port. The only exception is I2C (due to the EOMA-68 identification EEPROM): it is mandatory for all I/O Boards to provide an EOMA-68 identification EEPROM.

Exactly like legacy PCMCIA Cards, EOMA-68 CPU Cards may have absolutely any functions, any additional connectors, peripherals and so on *without* limitation, except for compliance with the EOMA-68 pinouts and physical size constraints. These additional functions, which may include access ports in the casework, may extend outwards from the user-facing end of the CPU Card to any practical extent, exactly as with legacy PCMCIA.

Background to Interface Selection

The interfaces have been specifically chosen because they are either essential or they are multi-purpose buses, and surprisingly they are perfectly adequate despite being Lowest Common Denominator across a wide range of CPUs for at least a decade. The goal here is not to attain ultra-high-speed latest-and-greatest performance but to use proven, long-established interfaces that will be easy to find parts for mass-volume appliances in potentially hundreds of millions of units and above.

Also, some graceful degradation through negotiation at the *hardware* level is not only desirable but is an essential distinctive and unique feature of the EOMA68 standard, because it dramatically simplifies the "sales pitch" as well as the engineering design, user card selection ("just get one of these, plug it into any product, it will work") and so on whilst at the same time ensuring that the range of SoCs that can be used is significantly diverse and future-proof.

- **I2C** - I2C is only two wires, is a global bus that can address multiple devices, and is a long-established proven Industry Standard with thousands of devices available.
- **USB** - USB2 is only two wires; USB3 is six. USB, like I2C, is a global bus that can address multiple devices and is a long-established proven Industry Standard.
- **Ethernet** - 10/100/1,000 Ethernet was chosen because it is prevalent on the majority of computing devices. In the case where chosen CPUs do not have Ethernet, a USB-to-Ethernet converter IC such as the SMSC LAN9500 can be deployed.
- **RGB/TTL** - 24-pin RGB/TTL was chosen over LVDS or MIPI so as to keep the cost down, and also to keep the signal speed down. Whilst LVDS seems initially to be a good candidate, Single-Channel LVDS is unsuitable for driving 1,920×1,080p60 LCD Panels: most 1,920×1,080 LCD panels require between 2 and 3 LVDS drivers. MIPI also requires multiple parallel channels to achieve higher data rates. Any low-cost CPU chosen which did not have LVDS or MIPI would be forced to add a converter chip, potentially on *both* sides of the interface (CPU card as well as motherboard). So it makes sense to choose the proven, lower-speed, reliable 24-pin interface, thus making the EOMA-68 Standard suitable for use even with ultra-low-cost 320×240 RGB/TTL LCD Panels, right the way up to HDTV screen sizes.
- **SD/MMC** - SD/MMC has a 4-pin, 2-pin, 1-pin and SPI mode. Transfer speed negotiation is possible at the hardware level. SPI can even be implemented as "bitbanging"

Note however in systems design that with the cost of mass-volume integrated SoCs dropping so significantly, the cost of deploying a SoC with USB-based (or other) peripheral ICs to make up for the lack of EOMA68-compliant interfaces (usually Ethernet or SATA) often *exceeds* the cost of competitor SoCs that *do* have the full complement of interfaces. An example is the Allwinner A20 (appx $7 in volume) vs e.g. a T.I $5 SoC or an Allwinner A13, neither of which have SATA or Ethernet. The addition of a USB-to-SATA Interface can add $1 to the BOM, whilst a USB-to-Ethernet IC can add $2. Given that a 4-port Hub IC would be required as well (an extra $1.50), it becomes clear that the A20 would win against the low-cost T.I SoC as well as against Allwinner's own low-cost A13 SoC on price, performance, reliability and power consumption.

Requirements for USB

All CPU Cards are required to support the full backwards-compatible auto-negotiation USB device capabilities and speeds of all former versions of the USB Interface, up to the maximum speed and capabilities chosen to be provided. Specifically:

- Providing USB Low Speed (version 1.0 - 1.5 Mbit/s) is acceptable.
- Providing USB Full Speed (version 1.1 - 12 Mbit/s) is acceptable if Low Speed is also provided.
- Providing USB Hi Speed (version 2.0 - 480 Mbit/s) is acceptable if Full Speed and Low Speed are also provided.
- Over the 1st USB port, providing USB Super Speed (version 3.0 - 5 Gbit/s) is acceptable if all lower speeds are also provided.
- Providing a higher version **only** and supporting no lower speeds is **not** acceptable.
- Providing no USB 3.0 is acceptable.

Chassis (devices) must support up to a maximum chosen USB specification and all speeds below. This guarantees that any CPU Card will work with any device, with any combination auto-negotiating to the maximum possible speed.

Requirements for Ethernet

All CPU Cards are required to support the full auto-negotiation capabilities of Ethernet, up to the maximum speed chosen to be provided. Specifically:

- Providing 10 Mbit/s Ethernet is acceptable
- Providing 100 Mbit/s Ethernet and down-negotiation to 10 Mbit/s Ethernet is acceptable
- Providing 100 Mbit/s Ethernet **only** is **not** acceptable
- Providing 1,000 Mbit/s Ethernet is acceptable as long as down-level negotiation to both 100 Mbit/s and 10 Mbit/s is also provided
- Providing 1,000 Mbit/s Ethernet **only** is **not** acceptable.

Chassis (devices) must also support up to a maximum chosen Ethernet specification and all speeds below. This guarantees that any CPU Card will work with any device, with any combination auto-negotiating to the maximum possible speed.

Requirements for RGB/TTL

The RGB/TTL output is the one point where close attention has to be paid on the part of the CPU Card designers, because of the variance between devices in which the CPU Cards will be plugged. This will need careful monitoring and may warrant a "Certification Programme" to ensure that CPU Cards are compliant with a wide range of devices.

- RGB/TTL is a parallel data bus, potentially running at up to 125 or even 150mhz. To ensure that the parallel signals are not skewed, *both* CPU Cards *and* I/O Boards **MUST** ensure that the length of the RGB/TTL tracks (data, HSYNC, VSYNC, CLK and DE) leading to the 68-pin connector - on either side of the 68-pin connector - are all of equal length. It is recommended that both the source (e.g the CPU) and the sink (e.g an LVDS IC) are placed as close to the 68-pin connector as possible.
- CPU Cards **must** provide software-programmable support for anywhere between 190x120 RGB-TTL resolutions all the way up to the maximum that they are capable of, with the maximum resolution being clearly marked on both the CPU Card, as well as the retail packaging in which it is sold.
- CPU Cards **should** support up to at least 1920x1080 at at least 50fps. However, some ultra-low-cost SoCs, especially those designed for mobile devices, only support up to XGA or WXGA resolutions. The use of such SoCs is not entirely recommended.
- EOMA-68's RGB/TTL interface is 24-bit-wide. If a particular SoC only has e.g. 18-bit or 15-bit RGB/TTL then the LSB (lower) bits **MUST** be set to logic output level 0 when the LCD interface is enabled: they must **NOT** be left floating or tri-state. This ensures that devices which are expecting the full 24-bits do not receive noise on the lower bits of each of the R,G and B 8-bit inputs.

Although there is no reason why individual devices should not have more than one LCD screen, allowing them to be selected, the burden of complexity for screen selection is placed onto the CPU Card software, so any company planning such a multi-screen device should contact the authors of the EOMA-68 specification (lkcl@qimod.com). Realistically, multi-

screen devices *should* consider instead using USB-based screen driver technology such as that from DisplayLink, or place any number of additional Display outputs onto the user-facing end of the CPU Card *(most CPU Cards will at least have a Micro-HDMI output)*.

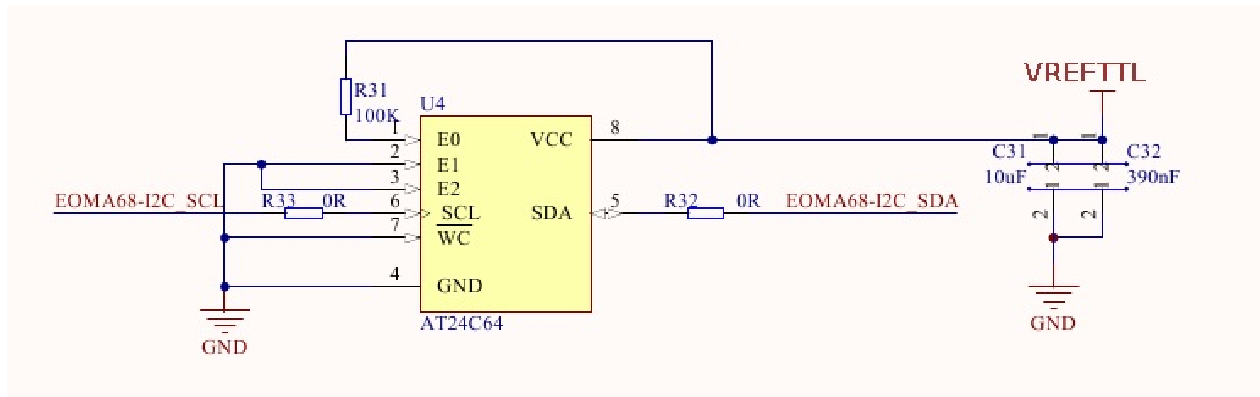
Requirements for I2C

These are the requirements for provision of I2C on an EOMA-68 interface. The summary is that the I2C bus must not be shared with any peripherals on the CPU Card, and the CPU Card must be able to read an on-board EEPROM (at address 0x51).

- The I2C bus that is connected to the EOMA-68 interface will expect to have access to an EEPROM that is addressable (readable) at I2C address 0x51.
- Additionally, there **MUST NOT** be *any* devices on the I2C bus on the CPU Card side. The reason is that all other addresses (other than 0x51) **must** be available for peripherals on the I/O Board.
- If a CPU Card needs to connect internally to any I2C peripherals on the PCB inside the CPU Card, the CPU Card **MUST** use a completely separate I2C bus (internally), **NOT** the one that is connected to the EOMA-68 Interface. i.e. the I2C bus that is connected to the EOMA-68 interface **MUST** remain completely dedicated to EOMA-68, and **MUST NOT** be shared with any peripherals on the CPU Card itself.
- The EEPROM **MUST NOT** be used for the storage of user data: it is reserved exclusively for EOMA-68.

Please note that there is considerable confusion over the definition of addresses in I2C. The discussion page has some clarification over what consititutes an address (7-bit) and what goes into the first 8 bits (7-bit address plus 1 bit indicating read or write). Adding to the confusion it is extremely common to find datasheets even from respectable companies that directly contradict the I2C specification.

Below is an example circuit showing an AT24C64 with the address set appropriately to 0x51. **PLEASE NOTE** that the AT24C64 datasheet **INCORRECTLY** misleads people to believe that the addresses are 0xA2 (for read) and 0xA3 (for write).

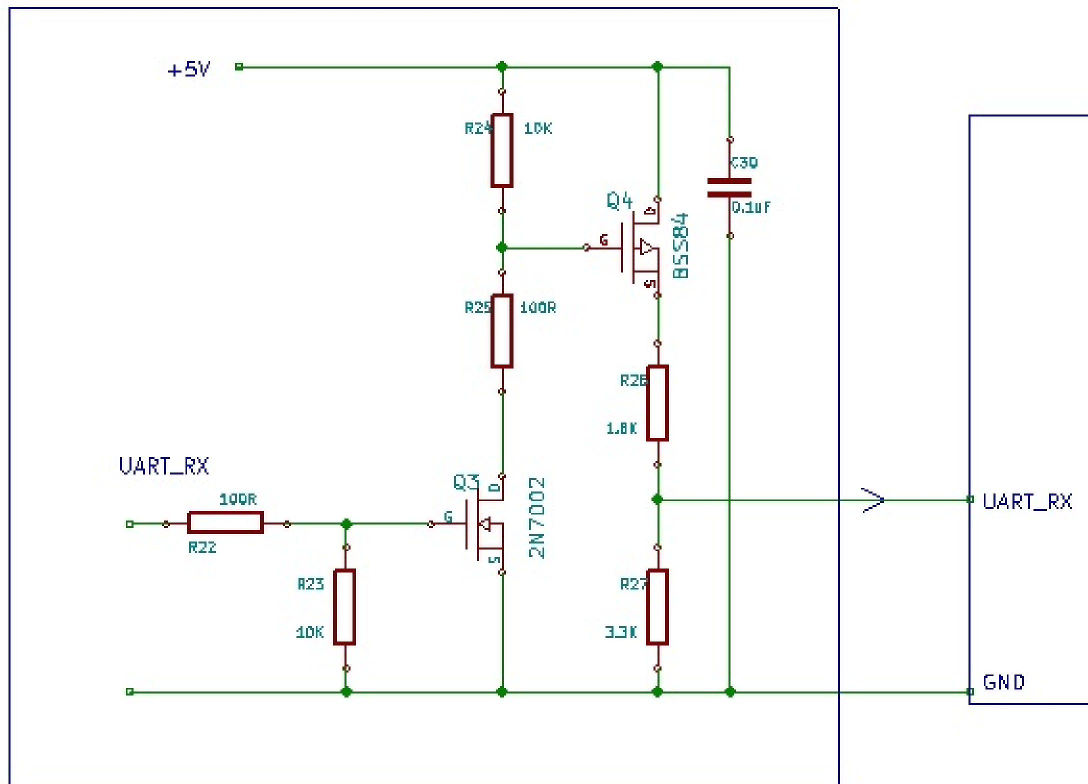


Requirements for UART

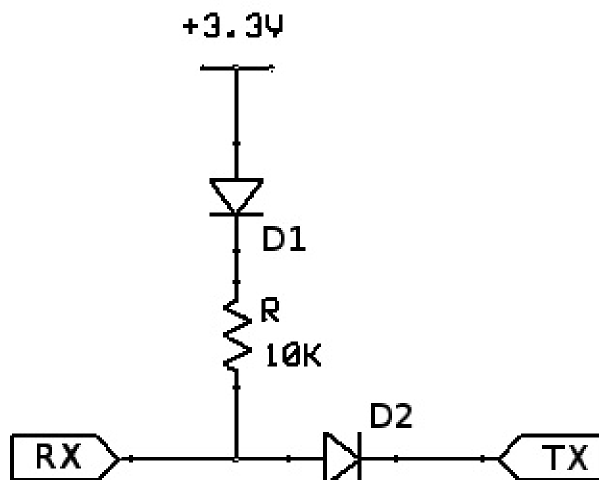
Strange as it may seem to have requirements for UART this section covers practical issues regarding protection of CPU Cards. When designing I/O Boards it is important to take into consideration that many embedded SoCs do not have proper UART buffering. Typically if the SoC is powered down but the I/O Board continues to be powered up such that it continues to provide a positive voltage to the UART "RX" line this can potentially result in power leakage through the SoC and on to other areas of the PCB. It is therefore critical that I/O Boards ensure that this does not happen.

As this problem is to be taken care of on the I/O Board it is worth observing that CPU Cards do not require UART buffering. They may however require level shifting: the signal levels are, like all other Digital I/O in EOMA68, expected to be 3.3v.

Below is an example circuit which can be used to protect the UART-RX line, using MOSFETs.



Here is another example that uses schottky diodes. D1 is to reduce the voltage slightly so that it will be below the 3.3v level that is internally supplied to the CPU Card. The same effect could reasonably be achieved using a resistor-divider bridge.



There are also other options such as the use of a MAX2322 RS232 buffer IC. Other options can be found here.

Requirements for SD/MMC and SPI

SD/MMC is a little strange in that it has hardware backwards-compatibility down to SPI in most controllers, but even if a hardware controller does not it is still possible to emulate SPI using "bitbanging". As bit-banging is quite CPU-intensive, and the transfer speed of SPI is 25mhz, designers of CPU Cards as well as designers of I/O Boards need to take this into consideration. It is therefore *recommended* that CPU Card designers either provide the full interoperable functionality (SD/MMC as well as SPI mode) or provide a means by which the hardware functionality is multiplex-routed to the 6 I/O pins whilst at the same time ensuring that the same 6 pins can be fully bi-directional as GPIO, (for example by using a small FPGA). As this latter option would be quite complex, it is best to provide the full functionality instead.

It is also critical to bear in mind that the pins **MUST** be shared (multiplexed) with bi-directional GPIO as noted in the EOMA-68 pinouts table. When designing the hardware, it must be taken into consideration that the option to switch all or any of the pins from GPIO to SD/MMC or SPI, including selecting 4-pin, 2-pin or 1-pin mode whilst the remaining unused pins are made available as GPIO, **MUST** be available at all times (i.e. not just as a boot-time option but dynamically at run-time).

For most SoCs these requirements are not burdensome: most SoCs already have multiplexed SD/MMC with bi-directional GPIO, where selection is programmable dynamically both at boot-time and run-time , and they support all modes of SD/MMC as well.

Pinouts (version 1.0)

These pinouts make no attempt to be electrically or electronically compatible with the legacy PCMCIA standard. 8 GPIO pins, 24-pin RGB/TTL, USB2, I2C, 10/100/1000 Ethernet and SATA-III interfaces are included in the Version 1.0 specification. *Note: USB2, SATA-III and Ethernet MUST support auto-negotiation, and MUST support the lower capabilities (USB 1, USB 1.1, SATA-I, SATA-II, 10/100 Ethernet). Higher speeds and capabilities are optional*.

Four 5.0 V power inputs must be provided: all pins are rated at 0.5 A, so the maximum power dissipation is limited to 10 watts. *Design consideration: please note that to ensure that thermal dissipation in an enclosed fanless situation is not exceeded, a maximum of 3.5 watts should be respected, or the card must contain its own fan (not recommended). Most systems will **not** have active cooling.*

All High-speed signals (USB2, Ethernet, SATA-III) are balanced lines that are still separated using GND or Power pins. All other pins are low frequency, with the exception of the LCD Pixel Clock and Pixel Data pins, which could go as high as 125 MHz for 1,920×1,080 @ 60fps (not recommended). The eight GPIO pins are available, for general-purpose bi-directional use of digital data only.

The output from the 24-pin LCD RGB/TTL pins must be electrically compatible with a Texas Instruments SN75LVDS83B, which has electrical characteristics of 3.3 V TTL but requires 5 V TTL tolerance. Typical TTL high-level voltage is 2.0 volts; threshold is 1.4 V; low-level TTL voltage is 0.8 V.

Also, because the GPIO pins can be reconfigured individually bi-directional for any digital purposes, they **must** be made to be 5 V TTL tolerant and tri-state isolated, and Motherboards also must be 5.0 V TTL tolerant as well as tri-state isolated. Levels when any GPIO pin is used either as an input or as an output should again operate at nominal 3.3 V TTL levels, thus expect "high" voltage of 2.0 volts, threshold of 1.4 V and "low" voltage of 0.8 V, but must accept voltages from 0–5 V.

The option for a CPU Card to provide Gigabit Ethernet is also available, if a given system has it. If, however, a particular system does not have Gigabit Ethernet, the pins **must not** be used for other purposes, and **must** be left unconnected (floating). This is to ensure that automatic negotiation of 100/1000 Ethernet occurs correctly.

The option for a CPU Card to provide USB3 is also available, if a given system has it. If, however, a particular system does not have USB3, the pins **must not** be used for other purposes, and **must** be left unconnected (floating). Additionally, I/O Boards **must not** use the unused pins for any other purpose and must leave them unconnected (floating). This is to ensure that automatic down-negotiation of USB2 occurs correctly and that damage does not occur to USB3-capable CPU Cards when plugged into I/O Boards with only USB2 capability.

Pin 22 is available for implementations to use for any special non-EOMA68-compliant purpose. I/O Boards **MUST NOT** rely on any specific card implementation providing any specific functionality on pin

1. Examples of appropriate uses for Pin 22 include start-up selection of a boot mode that is specific to a processor so that it is more convenient for a factory install to be able to re-flash and test an Operating System without needing to open up the case. Inappropriate uses are for example using Pin 22 as a 9th GPIO or as a One-Wire Bus, because not all I/O Boards will provide this exact same functionality.

Note also: for factory-install purposes, cards are of course permitted to use all and any pins, ports or methods required to carry out factory-installs and testing, as long as after factory-install the 68 pins are capable of EOMA-68 compliance. Examples of such uses would include a test-bench with an SD/MMC interface for first firmware boot, a JTAG interface and other diagnostics.

Table of EOMA-68 pinouts

Row 1	Row 2
* 1 GPIO (12) / SPI_MISO	* 35 GPIO (13) / SPI_MOSI
* 2 LCD Pixel Data bit 18 (Blue2)	* 36 LCD Pixel Data bit 19 (Blue3)
* 3 LCD Pixel Data bit 20 (Blue4)	* 37 LCD Pixel Data bit 21 (Blue5)
* 4 LCD Pixel Data bit 22 (Blue6)	* 38 LCD Pixel Data bit 23 (Blue7)
* 5 GPIO (14) / SPI_SCK	* 39 GPIO (15) / SPI_CS
* 6 LCD Pixel Data bit 10 (Green2)	* 40 LCD Pixel Data bit 11 (Green3)
* 7 LCD Pixel Data bit 12 (Green4)	* 41 LCD Pixel Data bit 13 (Green5)
* 8 LCD Pixel Data bit 14 (Green6)	* 42 LCD Pixel Data bit 15 (Green7)
* 9 GPIO (16) / EINT1	* 43 POWER#
* 10 LCD Pixel Data bit 2 (Red2)	* 44 LCD Pixel Data bit 3 (Red3)
* 11 LCD Pixel Data bit 4 (Red4)	* 45 LCD Pixel Data bit 5 (Red5)
* 12 LCD Pixel Data bit 6 (Red6)	* 46 LCD Pixel Data bit 7 (Red7)
* 13 LCD Pixel Clock	* 47 LCD Vertical Synchronization
* 14 LCD Horizontal Synchronization	* 48 LCD Pixel data enable (TFT) output
* 15 I2C Clock (SCL)	* 49 I2C Data (SDA)
* 16 GPIO (0) / SDMMC-D3	* 50 GPIO (1) / SDMMC-D2
* 17 GPIO (2)	* 51 GPIO (3)
* 18 GPIO (4) / SDMMC-CMD	* 52 GPIO (5) / SDMMC-CLK
* 19 GPIO (6) / SDMMC-D0	* 53 GPIO (7) / SDMMC-D1
* 20 ---- not used ---- / USB3 StdA_SSRX-	* 54 ---- not used ---- / USB3 StdA_SSRX+
* 21 ---- not used ---- / USB3 StdA_SSTX-	* 55 ---- not used ---- / USB3 StdA_SSTX+
* 22 GPIO (10) / PWM	* 56 Ethernet Feedback (transformer mid-point)
* 23 GPIO (8) / UART_TX	* 57 GPIO (9) / UART_RX
* 24 PWR (5.0 V)	* 58 PWR (5.0 V)
* 25 ---- not used ---- / 1000 Eth BI_DD+	* 59 ---- not used ---- / 1000 Eth BI_DD−
* 26 10/100 Ethernet (RX+) / 1000 Eth BI_DB+	* 60 10/100 Ethernet (RX−) / 1000 Eth BI_DB−
* 27 10/100 Ethernet (TX+) / 1000 Eth BI_DA+	* 61 10/100 Ethernet (TX−) / 1000 Eth BI_DA−
* 28 ---- not used ---- / 1000 Eth BI_DC+	* 62 ---- not used ---- / 1000 Eth BI_DC−
* 29 PWR (5.0 V)	* 63 PWR (5.0 V)
* 30 1st USB2 (Data+)	* 64 1st USB2 (Data−)
* 31 GROUND	* 65 GROUND
* 32 GPIO (11) / EINT0	* 66 VREF-TTL (GPIO TTL Voltage Reference)
* 33 GROUND	* 67 GROUND
* 34 2nd USB2 (Data+)	* 68 2nd USB2 (Data−)

Start-up procedure

It is required that all pins be disabled (floating tri-state) with the exception of the I2C Bus, the 5.0v Power and the Ground Pins. I2C Bus Master is then enabled, to search for an I2C EEPROM at address 0xA2. This EEPROM contains Linux Kernel "Device Tree" data, which specifies the devices available on the motherboard, as well as the actual pin-outs. The exact format of the EEPROM data is yet to be decided.

One important aspect of reading the I2C EEPROM is that the CPU card can then correctly access and initialise on-board devices. It also defines the purpose and use of the GPIO pins (if any are required). Also, the format of the LCD data is specified. For example, the pinout diagram on this page assumes 24-pin RGB TTL, but some motherboards may have LCD panels which only have an 18-pin RGB/TTL interface. The data in the I2C EEPROM therefore provides clear specifications on all the motherboard's peripherals.

Discussion of the startup procedure is here on arm-netbooks

Future Versions

All LCD and GPIO pins must be tri-state floating in order that future versions of this standard can provide faster (or merely alternative) interfaces. At the time of writing (2011), the interfaces in the 1.0 Specification are "Lowest Common Denominator" yet are still present across the majority of 2011's powerful embedded SoCs (OMAP4440, Enyxos4210, Tegra 3, iMX53, iMX6, Allwinner A10, A20 etc.) However, in the future, the "Lowest Common Denominator" could well comprise MIPI instead of RGB/TTL, 2 lane PCI-express (or its successor), and USB-3 instead of USB-2 (perhaps even a faster version of ULPI).

As of 2011 however, the total number of Embedded CPUs supporting all these newer interfaces and still keeping to a 1.5 watt budget is precisely zero. Support for these high-speed interfaces will therefore be re-evaluated in 2 to 3 years time, and a future version of this standard created when a large proportion of available embedded CPUs have these or other high-speed interfaces that are available at the time.

Deliberate Mechanical Non-interoperability

The re-use of the PCMCIA standard pinouts with no electrical or electronic compatibility requires mechanical means to ensure that newer cards cannot be inserted into legacy sockets. The proposed solution is therefore to deploy a fascia plate on the EOMA-68 card that is both larger in width than the standard 55 mm as well as recessed by some 8 mm, along the length of one of the 85 mm edges. The exact dimensions are yet to be determined, as specific PCMCIA housings need to be examined to ensure that one side can take the recessed "edge stop". The following part, PCMCIA Ejector Assembly from Tyco Electronics, is ideally suited: slimline and nothing at the left side.

Physical Dimensions

There are two sets of acceptable dimensions: as with the legacy PCMCIA interface, these must be backwards-compatible.

Type II

The physical dimensions are a maximum of "Type II" (i.e. 5mm maximum height). Cards should typically have all user-facing connectors "flush" with the standard PCMCIA size. This will ensure that when a Card is inserted into a device, the connectors of the CPU Card appear to be part of the actual device. However: devices should cater for the possibility that an EOMA-68 Card may have connectors sticking out of the end, to any practical height.

As the EOMA-68 pinouts have been designed to avoid matching the power lines of PCMCIA, there is no need for mechanical blocking.

Type III

Type III Cards have a maximum height of 8mm: this is typically reserved for x86-based CPUs which require up to 10 watts to operate. Motherboards that take the Type III cards **must** also accept the Type II lower-power cards.

TBC: Type III Cards should not assume that there will be fans available in the devices in which the cards are inserted, and should make arrangements for the removal of heat.

Thermal Considerations

In order to reduce the cost of Motherboards and system design, Type II Cards should not exceed an average of 3.5 watts power consumption for prolonged periods of time, despite there being provision for up to 10 watts on the EOMA-68 connector.

CPU Cards and Motherboards that support the Type III 8mm-high cards **must** be designed with a Thermal Dissipation capability that takes the 10 watt TDP into consideration, as well as taking into consideration the power consumption and heat generation of the devices used on the Motherboard as well. Whilst fan-based solutions are acceptable, the use of thermally-conductive copolymer plastics (some of which have thermal dissipation capabilities exceeding that of Aluminium) are recommended.

Header Connectors

Within the physical dimensions, there is absolutely no restriction on the number of connectors, interfaces, headers, expansion headers or antenna protruding from the end of the device. For example: a PCMCIA CPU card may typically have, for best useability, a Micro-HDMI, a USB-OTG, a 5-pin Audio Jack and a Micro-SD Card Slot. These four interfaces fit neatly within the 54 mm × 5.5 mm fascia plate size limit, as long as mid-mount connectors are used.

Also, on the actual EOMA-68 CPU Card PCB itself, there is no restriction on the number of expansion headers (populated or unpopulated) - the only consideration being that the EOMA-68 CPU card clearly cannot have expansion headers except for Engineers and Embedded Device Designers, and also have a metal shield installed around the EOMA-68 CPU card at the same time. However, there is no reason why the expansion headers should be unpopulated, supplied without a metal shield to Embedded Engineers, yet the exact same device shipped in mass-volume with a metal shield installed, for the average user.

The only issue to note is that there is a maximum power budget of about 10 watts (although there are four 5.0V 0.5A pins) but also that there is a practical maximum power dissipation of EOMA-68 cards of about 4 watts. There is no provision in the standard for air-cooling (fans) in the cases: most devices will be a passive-cooled layout.

If however the EOMA-68 card is designed to operate "stand-alone", for example by being provided with a Power Connector on its user-facing edge or by making use of USB-OTG, then of course the designers are free to disregard these constraints. If however the CPU card is also expected to operate inside a conformant device, then it must adjust accordingly and stick within the 4 watt heat dissipation budget.

Example Motherboards

Here is a list of example designs which conform the EOMA-68 Standard:

- Mini Engineering Board
  - suitable for Free Software Developers, ODM Development, SoC "Board Support Packages", Experimentation, Prototyping, Electrical Engineers, Training and R&D purposes.
- Monster Engineering Board
  - suitable for ODM Development "Demonstration" Purposes: designed to be "cut down to size", requiring the minimum amount of CAD/CAM Development effort and maximising return on investment.

- The Obligatory Tablet
  - a simple tablet motherboard which could potentially be developed as a very low cost single-sided 2-layer PCB. Components are chosen to reduce development cost and risk, as well as reduce manufacturing cost.
- Laptop
  - a laptop motherboard which could potentially be developed as a very low cost single-sided 2-layer PCB, through the use of modular and optional components for WIFI and 3G.
- LCD (TV)
  - an LCD Monitor (or Picture Frame) which can be upgraded into a TV or an All-in-One Computer or an Internet TV or Personal Video Recorder or Media Centre. very versatile yet simple to do.
- Passthrough or "Blank" Card
  - a special type of card which simply passes through the connectors, with little or no signal conversion.

Contact and Discussion

For questions, comments and general discussion, please use arm-netbook mailing list

Slides

Introduction of EOMA

FAQ

The FAQ is now on its own page.

Categories:

- Embedded Open Modular Architecture
- ARM processors

> From: eLinux.org

Flameman/routerboard-rb532

\< Flameman

For more interesting projects done by Flameman, be sure to check out his project index

Contents

RouterBoard-rb532-Flameman

Note

i'm developing for this board, the wiki page will be improved soon

feel free to contact me (see the contact below)

Introduction

The Target-goal of this page is

- install gentoo-mipsel into microdrive
- make the board able to boot from it

- describe how to build a jtag cable (to debug and recover from "Brickage")
- describe something useful with you can do with the board
- describe other Operating Systems avalaible for the board

logical steps about installing gentoo

- add the JTAG connector at J10 (you could skip it, it is suggested)
- build the JTAG cable (you could skip it, it is suggested))
- study the bootloader
- make partitions on the microdrive
- populate them
- set the bootloader environment to boot from the microdrive

People you could contact if you need help

- flameman, i'm currently use this board for a project, email
  - msn daredevil-coder@hotmail.it
  - email flamemaniii@gmail.com
  - irc.nick flameman (channel #edev, #gentoo-ppc)
- you ... if you want ;-)

About the board

rb532 is a shortened name for the routerboard

you could get your board from http://routerboard.com/rb500.html

see the nearest reseller of the area where you live mind a brand new boards costs 140Euro + shipping and VAT cheaper way requires (fortune and) ebay searched for

features

```
   * MIPS 32 4Kc based 266MHz (BIOS adjustable from 200 to 333 MHz; 400MHz processor factory option) embedded process
or
   * 32MB DDR onboard memory
   * Boot loader, RouterBOOT, 1Mbit Flash chip
   * Data storage, 128MB onboard
   * CompactFlash type I/II slot (also supports IBM/Hitachi Microdrive)
   * Two VIA VT6105, One IDT Korina, 10/100 Mbit/s Fast Ethernet
   * Two MiniPCI Type IIIA/IIIB slots
   * Daughterboard connector Present
   * One DB9 RS232C asynchronous serial port
   * LEDs. Power, 2 LED pairs for MiniPCI slots, 1 user LED
   * Watchdog IDT internal SoC hardware watchdog timer
   * IEEE802.3af Power over Ethernet: 12V or 48V DC Power jack: 11..60V DC
   * Power jack/header 6..22V or 25..56V DC jumper selectable.
   * Dimensions. 14.0 cm x 14.0 cm (5.51 in x 5.51 in)
   * Temperature. Operational: -20°C to +70°C (-4°F to 158°F)
   * Humidity. Operational: 70% relative humidity (non-condensing)
   * Currently supported OS. RouterOS 2.9, Linux 2.4, Linux 2.6
```

Overview

The board consists of:

- **CPU** IDT MIPS 79RC32434, a solid & smart implementation of the mipsel little endian MIPS 32 4Kcore CPU architecture, running at 400mhz (adjustable from 200 to 400MHz; 400MHz default and recommended)
- **RAM** soldered 64MB DDR onboard memory chip

- **LAN** On-chip 3 ethernet: One IDT Korina 10/100 Mbit/s Fast Ethernet port supporting Auto-MDI/X, Two VIA VT6105 10/100 Mbit/s Fast Ethernet ports supporting Auto-MDI/X.
- **UART** One DB9 RS232C asynchronous serial port, speeds up to 230k, tested up to 115200bps
- **miniPCI** Two MiniPCI Type IIIA/IIIB slots
- **CF** support for IBM/Hitachi Microdrive (http://eLinux.org/dev/cfa)
- **ROM** 1Mbit NAND flash where it is stored the bootloader
- **POWER** IEEE802.3af Power over Ethernet: 12V or 48V DC, Power jack/header 6..22V or 25..56V DC jumper selectable. PoE does not support power over datalines
- **System PCB 14.0 cm x 14.0 cm**
- **RTC** the real time clock chip is missing
- **LED** Power, 2 LED pairs for MiniPCI slots, 1 user LED
- **Watchdog** IDT internal SoC hardware watchdog timer

NOTE: from the base configuration of 3 Ethernet ports and 2 MiniPCI slots (for wireless cards) it can be extended by using daughterboards for four more MiniPCI slots, or even up to a total of 15 interfaces (9 Ethernet ports and 6 MiniPCI slots). A wide range of accepted input power means that you can connect to almost any direct current power source you already have on site. It also can be powered through an Ethernet port using Power-over-Ethernet technology (802.3af) or passive non-standard PoE.

NOTE2: there is a daughterboard for the RB532. It is called RouterBOARD 564, and It attaches to the RB532 by means of special daughterboard connector and adds four more MiniPCI slots and six ethernet ports. You can not use more than two AR5414 chipset (e.g. R52) cards in RB/564. If you want to use three or four miniPCI cards in RB/564, you should use other chipset based ones for those extra slots, like CM9. For example, two R52 + two CM9 is OK, four CM9 is also OK, but not three or four R52.

Memory Locations

memory map of the board will be added as soon as possible

addr begin	addr end	area
...	??	ram, userspace
400000	??	ram, userspace

Open questions

---

1) how/where is ram mapped ?

2) how/where is microdrive mapped ?

3) how/where is pci mapped ?

3) what is the bootstrap addr of the flash ?

...

Problems

...

Images of the board

also see a detailed imate @ www routerboard.com/img/rb500_l.jpg

Kernel

status

Korina ethernet VIA Rhine

Rhine-I-VT86C100A x Rhine-II-VT6102 Rhine-III-VT6105

version	host	target	toolchain	note
2.6.22 vanilla+patch	compiled on ppc-7410, minerva	arc=mips-el, target=mips-1	gcc-4.1.2-glibc binutils-2.19	rocks, 6 days uptime hardly compiling, korina sometime stalls, VIA works
2.6.24 vanilla+patch	compiled on ppc-7410, minerva	arc=mips-el, target=mips-1	gcc-4.1.2-glibc binutils-2.19	korina is now working, VIA works (korina and VIA statically compiled)
2.6.30.6 vanilla+patch --- REMOVED from kernel tree line ---	compiled on ppc-7410, minerva	arc=mips-el, target=mips-1	gcc-4.1.2-glibc binutils-2.19	korina is working, VIA is not working, name eth0 inversion, korina is not eth0, korina and VIA statically compiled, the suggestion looks like compiling the via-rhine driver as module and load it after the korina, this may works for 2.6.30.6,2.6.30.5,2.6.30.1 also with rc6
2.6.37 vanilla+patch	compiled on ppc-7550, vittoria	arc=mips-el, target=mips-1	gcc-4.1.2-glibc binutils-2.19	korina is working, VIA is working, intel-ipw2200(wifi minipci) is working (with an hack), cmdline patched, kernel-panik() has been patched

download

kernel 2.6.22 full tested and working kernel-gentoo-rb532-2.6.22.gz (suggested for production)

rootfs

dev

cfa

mknod /dev/cfa b 13 0 mknod /dev/cfa1 b 13 1 mknod /dev/cfa2 b 13 2 mknod /dev/cfa3 b 13 3

gpio

sources

http://tomoyo.sourceforge.jp/cgi-bin/lxr/source/arch/mips/rb532/gpio.c

gpio.c

```
arch.mips.rb532.gpio

001 /*
002  *  Miscellaneous functions for IDT EB434 board
003  *
004  *  2004 IDT Inc. (rischelp@idt.com)
005  *  2006 Phil Sutter <n0-1@freewrt.org>
006  *  2007 Florian Fainelli <florian@openwrt.org>
007  *
027  */
028
029 #include <linux/kernel.h>
030 #include <linux/gpio.h>
031 #include <linux/init.h>
032 #include <linux/types.h>
033 #include <linux/pci.h>
034 #include <linux/spinlock.h>
035 #include <linux/io.h>
```

```
036 #include <linux/platform_device.h>
037
038 #include <asm/addrspace.h>
039
040 #include <asm/mach-rc32434/rb.h>
041
042 struct rb532_gpio_reg __iomem *rb532_gpio_reg0;
043 EXPORT_SYMBOL(rb532_gpio_reg0);
044
045 struct mpmc_device dev3;
046
047 static struct resource rb532_gpio_reg0_res[] = {
048         {
049                 .name   = "gpio_reg0",
050                 .start  = (u32)(IDT434_REG_BASE + GPIOBASE),
051                 .end    = (u32)(IDT434_REG_BASE + GPIOBASE + sizeof(struct rb532_gpio_reg)),
052                 .flags  = IORESOURCE_MEM,
053         }
054 };
055
056 static struct resource rb532_dev3_ctl_res[] = {
057         {
058                 .name   = "dev3_ctl",
059                 .start  = (u32)(IDT434_REG_BASE + DEV3BASE),
060                 .end    = (u32)(IDT434_REG_BASE + DEV3BASE + sizeof(struct dev_reg)),
061                 .flags  = IORESOURCE_MEM,
062         }
063 };
064
065 void set_434_reg(unsigned reg_offs, unsigned bit, unsigned len, unsigned val)
066 {
067         unsigned long flags;
068         unsigned data;
069         unsigned i = 0;
070
071         spin_lock_irqsave(&dev3.lock, flags);
072
073         data = *(volatile unsigned *) (IDT434_REG_BASE + reg_offs);
074         for (i = 0; i != len; ++i) {
075                 if (val & (1 << i))
076                         data |= (1 << (i + bit));
077                 else
078                         data &= ~(1 << (i + bit));
079         }
080         writel(data, (IDT434_REG_BASE + reg_offs));
081
082         spin_unlock_irqrestore(&dev3.lock, flags);
083 }
084 EXPORT_SYMBOL(set_434_reg);
085
086 unsigned get_434_reg(unsigned reg_offs)
087 {
088         return readl(IDT434_REG_BASE + reg_offs);
089 }
090 EXPORT_SYMBOL(get_434_reg);
091
092 void set_latch_u5(unsigned char or_mask, unsigned char nand_mask)
093 {
094         unsigned long flags;
095
096         spin_lock_irqsave(&dev3.lock, flags);
097
098         dev3.state = (dev3.state | or_mask) & ~nand_mask;
099         writel(dev3.state, &dev3.base);
100
101         spin_unlock_irqrestore(&dev3.lock, flags);
102 }
103 EXPORT_SYMBOL(set_latch_u5);
104
105 unsigned char get_latch_u5(void)
106 {
107         return dev3.state;
```

```
108 }
109 EXPORT_SYMBOL(get_latch_u5);
110
111 int rb532_gpio_get_value(unsigned gpio)
112 {
113         return readl(&rb532_gpio_reg0->gpiod) & (1 << gpio);
114 }
115 EXPORT_SYMBOL(rb532_gpio_get_value);
116
117 void rb532_gpio_set_value(unsigned gpio, int value)
118 {
119         unsigned tmp;
120
121         tmp = readl(&rb532_gpio_reg0->gpiod) & ~(1 << gpio);
122         if (value)
123                 tmp |= 1 << gpio;
124
125         writel(tmp, (void *)&rb532_gpio_reg0->gpiod);
126 }
127 EXPORT_SYMBOL(rb532_gpio_set_value);
128
129 int rb532_gpio_direction_input(unsigned gpio)
130 {
131         writel(readl(&rb532_gpio_reg0->gpiocfg) & ~(1 << gpio),
132                (void *)&rb532_gpio_reg0->gpiocfg);
133
134         return 0;
135 }
136 EXPORT_SYMBOL(rb532_gpio_direction_input);
137
138 int rb532_gpio_direction_output(unsigned gpio, int value)
139 {
140         gpio_set_value(gpio, value);
141         writel(readl(&rb532_gpio_reg0->gpiocfg) | (1 << gpio),
142                (void *)&rb532_gpio_reg0->gpiocfg);
143
144         return 0;
145 }
146 EXPORT_SYMBOL(rb532_gpio_direction_output);
147
148 void rb532_gpio_set_int_level(unsigned gpio, int value)
149 {
150         unsigned tmp;
151
152         tmp = readl(&rb532_gpio_reg0->gpioilevel) & ~(1 << gpio);
153         if (value)
154                 tmp |= 1 << gpio;
155         writel(tmp, (void *)&rb532_gpio_reg0->gpioilevel);
156 }
157 EXPORT_SYMBOL(rb532_gpio_set_int_level);
158
159 int rb532_gpio_get_int_level(unsigned gpio)
160 {
161         return readl(&rb532_gpio_reg0->gpioilevel) & (1 << gpio);
162 }
163 EXPORT_SYMBOL(rb532_gpio_get_int_level);
164
165 void rb532_gpio_set_int_status(unsigned gpio, int value)
166 {
167         unsigned tmp;
168
169         tmp = readl(&rb532_gpio_reg0->gpioistat);
170         if (value)
171                 tmp |= 1 << gpio;
172         writel(tmp, (void *)&rb532_gpio_reg0->gpioistat);
173 }
174 EXPORT_SYMBOL(rb532_gpio_set_int_status);
175
176 int rb532_gpio_get_int_status(unsigned gpio)
177 {
178         return readl(&rb532_gpio_reg0->gpioistat) & (1 << gpio);
179 }
```

```
180 EXPORT_SYMBOL(rb532_gpio_get_int_status);
181
182 void rb532_gpio_set_func(unsigned gpio, int value)
183 {
184         unsigned tmp;
185
186         tmp = readl(&rb532_gpio_reg0->gpiofunc);
187         if (value)
188                 tmp |= 1 << gpio;
189         writel(tmp, (void *)&rb532_gpio_reg0->gpiofunc);
190 }
191 EXPORT_SYMBOL(rb532_gpio_set_func);
192
193 int rb532_gpio_get_func(unsigned gpio)
194 {
195         return readl(&rb532_gpio_reg0->gpiofunc) & (1 << gpio);
196 }
197 EXPORT_SYMBOL(rb532_gpio_get_func);
198
199 int __init rb532_gpio_init(void)
200 {
201         rb532_gpio_reg0 = ioremap_nocache(rb532_gpio_reg0_res[0].start,
202                                 rb532_gpio_reg0_res[0].end -
203                                 rb532_gpio_reg0_res[0].start);
204
205         if (!rb532_gpio_reg0) {
206                 printk(KERN_ERR "rb532: cannot remap GPIO register 0\n");
207                 return -ENXIO;
208         }
209
210         dev3.base = ioremap_nocache(rb532_dev3_ctl_res[0].start,
211                                 rb532_dev3_ctl_res[0].end -
212                                 rb532_dev3_ctl_res[0].start);
213
214         if (!dev3.base) {
215                 printk(KERN_ERR "rb532: cannot remap device controller 3\n");
216                 return -ENXIO;
217         }
218
219         return 0;
220 }
221 arch_initcall(rb532_gpio_init);
```

About Gentoo

the kernel bootstrap from the micro drive



what/where

- **kernel 2.4.x** there is a port of linux 2.4.30/31 that has been tested and it is perfectly working: see http://routerboard.com/files/linux-2.4.30-yaffs2.patch.gz , http://routerboard.com/files/linux-2.4.31.patch.gz
- **kernel 2.6.x**
    - 2.6.22: in use on my board with success, and has it has been tested for a very long uptime it is known to be very robust (suggested for production)
    - 2.6.23: is under porting, see the git @ http://www.linux-mips.org/git?p=linux-routerboard.git;a=summary (actually i have an issue and it doesn't boot ... there is an issue with the firmware bootloader)
    - 2.6.24: no sup, repository romoved cause it still has the 2.6.23 issue
    - 2.6.26: interesting for the thickness feature, r3 is under development
    - 2.6.28: interesting for ext4, it's in my repository, r4 under development

dmesg

on 04-12-2007 i was able to compile a perfectly working kernel 2.6.22, here the dmesg

```
Version 2.6.22-manatnees-batman-mipsel-rb532 (root@queen-vittoria)
(gcc version 4.1.2 (Gentoo 4.1.2 p1.0.2)) #6 Tue Dec 4 17:12:23 CET 2007
CPU revision is: 0001800a
Determined physical RAM map:
 memory: 03fffa00 @ 00000400 (usable)
Wasting 32 bytes for tracking 1 unused pages
Initrd not found or empty - disabling initrd
On node 0 totalpages: 16383
  Normal zone: 127 pages used for memmap
  Normal zone: 0 pages reserved
  Normal zone: 16256 pages, LIFO batch:3
Built 1 zonelists.  Total pages: 16256
Kernel command line: root=/dev/cfa3 console=ttyS0,9600
gpio=8191 kmac=00:0C:42:0E:8F:01 board=500r5 boot=1
```

```
korina mac = 00:0C:42:0E:8F:01
Primary instruction cache 8kB, physically tagged, 4-way, linesize 16 bytes.
Primary data cache 8kB, 4-way, linesize 16 bytes.
Synthesized TLB refill handler (20 instructions).
Synthesized TLB load handler fastpath (32 instructions).
Synthesized TLB store handler fastpath (32 instructions).
Synthesized TLB modify handler fastpath (31 instructions).
Initializing IRQ's: 168 out of 256
PID hash table entries: 256 (order: 8, 1024 bytes)
calculating r4koff... 001e846c(1999980)
CPU frequency 400.00 MHz
Using 199.998 MHz high precision timer.
Dentry cache hash table entries: 8192 (order: 3, 32768 bytes)
Inode-cache hash table entries: 4096 (order: 2, 16384 bytes)
Memory: 61120k/65528k available
(2187k kernel code, 4348k reserved, 352k data, 120k init, 0k highmem)
Calibrating delay loop... 398.95 BogoMIPS (lpj=1994752)
Mount-cache hash table entries: 512
NET: Registered protocol family 16
PCI: Initializing PCI
registering PCI controller with io_map_base unset
NET: Registered protocol family 2
Time: MIPS clocksource has been installed.
IP route cache hash table entries: 1024 (order: 0, 4096 bytes)
TCP established hash table entries: 2048 (order: 2, 16384 bytes)
TCP bind hash table entries: 2048 (order: 1, 8192 bytes)
TCP: Hash tables configured (established 2048 bind 2048)
TCP reno registered
Registering mini_fo version $Id$
JFFS2 version 2.2. (NAND) (SUMMARY)  Â© 2001-2006 Red Hat, Inc.
yaffs Dec  4 2007 17:07:52 Installing.
io scheduler noop registered
io scheduler deadline registered (default)
Serial: 8250/16550 driver $Revision: 1.90 $ 2 ports, IRQ sharing disabled
serial8250: ttyS0 at MMIO 0x0 (irq = 104) is a 16550A
cf-mips module loaded
cf-mips: resetting..
cf-mips: identify drive..
cf-mips: CF card detected, C/H/S=3968/16/63 sectors=3999744 (1953MB)
cf-mips: detecting block size
cf-mips: multiple sectors = 32
init done: cfa: cfa1 cfa2 cfa3
Using NAPI with weight 64
eth0: Rx IRQ 40, Tx IRQ 41, 00:0c:42:0e:8f:01
via-rhine.c:v1.10-LK1.4.3 2007-03-06 Written by Donald Becker
PCI: Enabling device 0000:00:02.0 (0080 -> 0083)
PCI: Setting latency timer of device 0000:00:02.0 to 64
io_map_base of root PCI bus 0000:00 unset.  Trying to continue but you better
fix this issue or report it to linux-mips@linux-mips.org or your vendor.
eth1: VIA Rhine III at 0xb8800000, 00:0c:42:0e:8f:02, IRQ 142.
eth1: MII PHY found at address 1, status 0x7849 advertising 05e1 Link 0000.
PCI: Enabling device 0000:00:03.0 (0080 -> 0083)
PCI: Setting latency timer of device 0000:00:03.0 to 64
eth2: VIA Rhine III at 0xb8800100, 00:0c:42:0e:8f:03, IRQ 143.
eth2: MII PHY found at address 1, status 0x7849 advertising 05e1 Link 0000.
block2mtd: version $Revision: 1.30 $
NAND device: Manufacturer ID: 0xad, Chip ID: 0xf1 (Hynix NAND 128MiB 3,3V 8-bit)
Scanning device for bad blocks
Bad eraseblock 92 at 0x00b80000
Creating 2 MTD partitions on "NAND 128MiB 3,3V 8-bit":
0x00000000-0x00400000 : "Routerboard NAND boot"
0x00400000-0x08000000 : "rootfs"
mtd: partition "rootfs" set to be root filesystem
split_squasfs: no squashfs found in "NAND 128MiB 3,3V 8-bit"
Registered led device: rb500led:amber
nf_conntrack version 0.5.0 (511 buckets, 4088 max)
ip_tables: (C) 2000-2006 Netfilter Core Team
TCP vegas registered
NET: Registered protocol family 1
NET: Registered protocol family 17
802.1Q VLAN Support v1.8 Ben Greear <greearb@candelatech.com>
All bugs added by David S. Miller <davem@redhat.com>
```

```
EXT3-fs: INFO: recovery required on readonly filesystem.
EXT3-fs: write access will be enabled during recovery.
kjournald starting.  Commit interval 5 seconds
EXT3-fs: recovery complete.
EXT3-fs: mounted filesystem with ordered data mode.
VFS: Mounted root (ext3 filesystem) readonly.
Freeing unused kernel memory: 120k freed
Algorithmics/MIPS FPU Emulator v1.5
EXT3 FS on cfa3, internal journal
```

@15-02-2009 i have to update the kernel to 2.6.26/2.6.28 (still under proof status in my repositories) and i am looking for {miniPCI-sATA, miniPCI-netwifi}

About devtools

@16-02-2009

you can ask me a working kernel tarball of the sources

Project

Router 3 port ethernet

coming soon

Category:

- Mips

> From: eLinux.org

Freescale IMX53QSB

Contents

Topics

Hardware

The i.MX53 Quick Start Board (aka LOCO) is a 1 GHz ARM Cortex-A8 embedded computer on a 3-inch by 3-inch board.

The loco boots from SD Card with the boot loader at the first sector of it.

You can use Barebox on it:

http://git.pengutronix.de/?p=barebox.git;a=summary

Barebox

Barebox uses the same build (Kbuild) and configuration (Kconfig) tools as the Linux kernel.

1) First you need to clone the tree

```
git clone git://git.pengutronix.de/git/barebox.git
```

Most probably you want to use a released Barebox version, by running 'git checkout \'. Check which versions are available with 'git tag -l' and use the latest one.

2) Then you need to configure it:

```
make freescale_mx53_loco_defconfig
```

3) Compile it

```
make
```

4) Now you need to generate your SD Card:

Plug the SD Card into your laptop and copy barebox.bin to it:

```
dd if=barebox.bin of=/dev/sdb bs=512
sync
```

5) plug the SD card into the board and boot it (press the boot button)

Now that you have flash, you will see Barebox booting and trying to start the kernel from the SD card.

```
barebox 2011.09.0-00338-ga6d06f2 (Oct  9 2011 - 23:07:34)

Board: Freescale i.MX53 LOCO
registered netconsole as cs1
eth@eth0: got MAC address from EEPROM: 00:04:9F:01:AB:1D
Malloc space: 0x7df00000 -> 0x7ff00000 (size 32 MB)
Stack space : 0x7def8000 -> 0x7df00000 (size 32 kB)
envfs: wrong magic on /dev/env0
no valid environment found on /dev/env0. Using default environment
running /env/bin/init...
barebox@Freescale i.MX53 LOCO:/
```

Kernel

You can use the mainline kernel, but take into account that not all IP cores available in the MX53 are already supported there. It is also possible to use Freescale's kernel, which can be found here:

http://opensource.freescale.com/git?p=imx/linux-2.6-imx.git;a=summary

1) First you need to clone the tree:

a) Mainline

```
git clone git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux-2.6.git
```

Note that, like with Barebox, if you aim for stability, you should take a released kernel revision instead of some random version from git.

b) Freescale

```
git clone http://opensource.freescale.com/pub/scm/imx/linux-2.6-imx.git

git checkout -b work imx_2.6.38_11.09.01
```

or

To speed up the clone, first clone the linus tree, then fetch freescale as this

or use any already cloned kernel

```
git clone git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux-2.6.git

git remote add freescale http://opensource.freescale.com/pub/scm/imx/linux-2.6-imx.git

git fetch freescale

git checkout -b work freescale/imx_2.6.38_11.09.01
```

2) Then you need to configure the kernel:

```
make imx5_defconfig
```

3) Compile it

```
make
```

If you have a machine with more than one core, add -j \.

4) now you need to generate your SD Card:

Plug the sd card on your laptop and copy barebox.bin on it

```
dd if=arch/arm/boot/uImage of=/dev/sdb bs=512 seek=768
sync
```

5) Plug the SD card into the board and boot it (press the boot button)

Now that you have flashed everything, you will see barebox booting and trying to start the kernel from the SD card.

```
barebox 2011.09.0-00338-ga6d06f2 (Oct  9 2011 - 23:07:34)

Board: Freescale i.MX53 LOCO
registered netconsole as cs1
eth@eth0: got MAC address from EEPROM: 00:04:9F:01:AB:1D
Malloc space: 0x7df00000 -> 0x7ff00000 (size 32 MB)
Stack space : 0x7def8000 -> 0x7df00000 (size 32 kB)
envfs: wrong magic on /dev/env0
no valid environment found on /dev/env0. Using default environment
running /env/bin/init...

Enter the Password within 3s to stop the boot
Password: booting kernel of type uimage from /dev/disk0.kernel
  Verifying Checksum ... OK
  Image Name:   Linux-2.6.35.3-00745-gce4c61a-di
  Created:      2011-10-09  13:43:29 UTC
  Image Type:   ARM Linux Multi-File Image (uncompressed)
  Data Size:    2885707 Bytes =  2.8 MB
  Load Address: 70008000
  Entry Point:  70008000
  Contents:
     Image 0: 2729640 Bytes =  2.6 MB
     Image 1: 156067 Bytes = 152.4 kB
      Offset = 0x7e5d0708
use initrd @1
initrd_start=0x72000000
OK

Starting kernel with initrd ...

commandline: console=ttymxc0,115200 ip=192.168.201.33:192.168.201.98:192.168.201.98:255.255.255.0::eth0: root=/dev/ra
m0 rdinit=/sbin/init
arch_number: 3273
Linux version 2.6.35.3-00745-gce4c61a-dirty (root@j-debian) (gcc version 4.5.1 (Sourcery G++ Lite 2010.09-50) ) #11 P
REEMPT Sat Oct 8 21:35:40 CST 2011
CPU: ARMv7 Processor [412fc085] revision 5 (ARMv7), cr=10c53c7f
CPU: VIPT nonaliasing data cache, VIPT nonaliasing instruction cache
Machine: Freescale MX53 LOCO Board
Memory policy: ECC disabled, Data cache writeback
Built 1 zonelists in Zone order, mobility grouping on.  Total pages: 218112
Kernel command line: console=ttymxc0,115200 ip=192.168.201.33:192.168.201.98:192.168.201.98:255.255.255.0::eth0: root
=/dev/ram0 rdinit=/sbin/init
PID hash table entries: 4096 (order: 2, 16384 bytes)
Dentry cache hash table entries: 131072 (order: 7, 524288 bytes)
Inode-cache hash table entries: 65536 (order: 6, 262144 bytes)
Memory: 352MB 512MB = 864MB total
Memory: 868548k/868548k available, 16188k reserved, 0K highmem
Virtual kernel memory layout:
   vector  : 0xffff0000 - 0xffff1000   (   4 kB)
   fixmap  : 0xfff00000 - 0xfffe0000   ( 896 kB)
   DMA     : 0xf9e00000 - 0xffe00000   (  96 MB)
   vmalloc : 0xe0800000 - 0xf4000000   ( 312 MB)
   lowmem  : 0x80000000 - 0xe0000000   (1536 MB)
   pkmap   : 0x7fe00000 - 0x80000000   (   2 MB)
   modules : 0x7f000000 - 0x7fe00000   (  14 MB)
     .init : 0x80008000 - 0x8002f000   ( 156 kB)
     .text : 0x8002f000 - 0x8078b000   (7536 kB)
     .data : 0x807ac000 - 0x807f3aa0   ( 287 kB)
SLUB: Genslabs=11, HWalign=32, Order=0-3, MinObjects=0, CPUs=1, Nodes=1
Hierarchical RCU implementation.
       RCU-based detection of stalled CPUs is disabled.
```

```
       Verbose stalled-CPUs detection is disabled.
NR_IRQS:368
MXC GPIO hardware
MXC IRQ initialized
MXC_Early serial console at MMIO 0x53fbc000 (options '115200')
bootconsole [ttymxc0] enabled
Console: colour dummy device 80x30
Calibrating delay loop... 999.42 BogoMIPS (lpj=4997120)
pid_max: default: 32768 minimum: 301
Mount-cache hash table entries: 512
CPU: Testing write buffer coherency: ok
devtmpfs: initialized
regulator: core version 0.5
NET: Registered protocol family 16
i.MX IRAM pool: 128 KB@0xe0840000
IRAM READY
CPU is i.MX0 Revision 0.0
Using SDMA I.API
MXC DMA API initialized
IMX usb wakeup probe
IMX usb wakeup probe
bio: create slab <bio-0> at 0
SCSI subsystem initialized
usbcore: registered new interface driver usbfs
usbcore: registered new interface driver hub
usbcore: registered new device driver usb
da9052_i2c_is_connected - i2c read success.............
regulator: DA9052_LDO1: 600 <--> 1800 mV at 1300 mV normal
regulator: DA9052_LDO2: 600 <--> 1800 mV at 1300 mV normal
regulator: DA9052_LDO3: 1725 <--> 3300 mV at 3300 mV normal
regulator: DA9052_LDO4: 1725 <--> 3300 mV at 2775 mV normal
regulator: DA9052_LDO5: 1200 <--> 3600 mV at 1300 mV normal
regulator: DA9052_LDO6: 1200 <--> 3600 mV at 1300 mV normal
regulator: DA9052_LDO7: 1200 <--> 3600 mV at 2750 mV normal
regulator: DA9052_LDO8: 1200 <--> 3600 mV at 1800 mV normal
regulator: DA9052_LDO9: 1250 <--> 3650 mV at 1500 mV normal
regulator: DA9052_LDO10: 1200 <--> 3600 mV at 1300 mV normal
regulator: DA9052_BUCK_CORE: 500 <--> 2075 mV at 1100 mV normal
regulator: DA9052_BUCK_PRO: 500 <--> 2075 mV at 1300 mV normal
regulator: DA9052_BUCK_MEM: 925 <--> 2500 mV at 1500 mV normal
regulator: DA9052_BUCK_PERI: 1800 <--> 3600 mV at 3600 mV normal
IPU DMFC NORMAL mode: 1(0~1), 5B(4,5), 5F(6,7)
Advanced Linux Sound Architecture Driver Version 1.0.23.
Bluetooth: Core ver 2.15
NET: Registered protocol family 31
Bluetooth: HCI device and connection manager initialized
Bluetooth: HCI socket layer initialized
Switching to clocksource mxc_timer1
NET: Registered protocol family 2
IP route cache hash table entries: 32768 (order: 5, 131072 bytes)
TCP established hash table entries: 131072 (order: 8, 1048576 bytes)
TCP bind hash table entries: 65536 (order: 6, 262144 bytes)
TCP: Hash tables configured (established 131072 bind 65536)
TCP reno registered
UDP hash table entries: 512 (order: 1, 8192 bytes)
UDP-Lite hash table entries: 512 (order: 1, 8192 bytes)
NET: Registered protocol family 1
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
RPC: Registered tcp NFSv4.1 backchannel transport module.
Unpacking initramfs...
Freeing initrd memory: 152K
LPMode driver module loaded
Static Power Management for Freescale i.MX5
PM driver module loaded
sdram autogating driver module loaded
Bus freq driver module loaded
DI0 is primary
mxc_dvfs_core_probe
DVFS driver module loaded
i.MXC CPU frequency driver
DVFS PER driver module loaded
```

```
squashfs: version 4.0 (2009/01/31) Phillip Lougher
msgmni has been set to 1696
alg: No test for stdrng (krng)
cryptodev: driver loaded.
io scheduler noop registered
io scheduler deadline registered
io scheduler cfq registered (default)
mxc_ipu mxc_ipu: Channel already disabled 9
mxc_ipu mxc_ipu: Channel already uninitialized 9
Console: switching to colour frame buffer device 100x30
tve: probe of tve.0 failed with error -1
sii9022 1-0039: SII9022: cound not find device
Serial: MXC Internal UART driver
mxcintuart.0: ttymxc0 at MMIO 0x53fbc000 (irq = 31) is a Freescale i.MX
console [ttymxc0] enabled, bootconsole disabled
console [ttymxc0] enabled, bootconsole disabled
mxcintuart.1: ttymxc1 at MMIO 0x53fc0000 (irq = 32) is a Freescale i.MX
mxcintuart.2: ttymxc2 at MMIO 0x5000c000 (irq = 33) is a Freescale i.MX
mxcintuart.3: ttymxc3 at MMIO 0x53ff0000 (irq = 13) is a Freescale i.MX
mxcintuart.4: ttymxc4 at MMIO 0x63f90000 (irq = 86) is a Freescale i.MX
loop: module loaded
ahci: SSS flag set, parallel bus scan disabled
ahci ahci.0: AHCI 0001.0100 32 slots 1 ports 3 Gbps 0x1 impl platform mode
ahci ahci.0: flags: ncq sntf stag pm led clo only pmp pio slum part ccc
scsi0 : ahci
ata1: SATA max UDMA/133 mmio [mem 0x10000000-0x10000fff] port 0x100 irq 28
MXC MTD nand Driver 3.0
vcan: Virtual CAN interface driver
Freescale FlexCAN Driver
FEC Ethernet Driver
fec_enet_mii_bus: probed
ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
fsl-ehci fsl-ehci.0: Freescale On-Chip EHCI Host Controller
fsl-ehci fsl-ehci.0: new USB bus registered, assigned bus number 1
fsl-ehci fsl-ehci.0: irq 18, io base 0x53f80000
fsl-ehci fsl-ehci.0: USB 2.0 started, EHCI 1.00
hub 1-0:1.0: USB hub found
hub 1-0:1.0: 1 port detected
fsl-ehci fsl-ehci.1: Freescale On-Chip EHCI Host Controller
fsl-ehci fsl-ehci.1: new USB bus registered, assigned bus number 2
ata1: SATA link down (SStatus 0 SControl 300)
fsl-ehci fsl-ehci.1: irq 14, io base 0x53f80200
fsl-ehci fsl-ehci.1: USB 2.0 started, EHCI 1.00
hub 2-0:1.0: USB hub found
hub 2-0:1.0: 1 port detected
Initializing USB Mass Storage driver...
usbcore: registered new interface driver usb-storage
USB Mass Storage support registered.
usbcore: registered new interface driver usbserial
usbserial: USB Serial Driver core
USB Serial support registered for GSM modem (1-port)
usbcore: registered new interface driver option
option: v0.7.2:USB Driver for GSM modems
mice: PS/2 mouse device common for all mice
input: gpio-keys as /devices/platform/gpio-keys/input/input0
MXC keypad loaded
DA9052 TSI Device Driver, v1.0
input: da9052-tsi as /devices/virtual/input/input1
TSI Drv Successfully Inserted da9052-tsi
input: da9052-onkey as /devices/platform/imx-i2c.0/i2c-0/0-0048/da9052-onkey/input/input2
mxc_rtc mxc_rtc.0: rtc core: registered mxc_rtc as rtc0
i2c /dev entries driver
IR NEC protocol handler initialized
IR RC5(x) protocol handler initialized
IR RC6 protocol handler initialized
IR JVC protocol handler initialized
IR Sony protocol handler initialized
Linux video capture interface: v2.00
mxc_v4l2_output mxc_v4l2_output.0: Registered device video0
usbcore: registered new interface driver uvcvideo
USB Video Class driver (v0.1.0)
APM Battery Driver
```

```
check mma8450 chip ID
mma8450 0-001c: build time Oct  6 2011 02:19:40
input: mma8450 as /devices/platform/imx-i2c.0/i2c-0/0-001c/input/input3
add mma8450 i2c driver
add mma8451 i2c driver
MXC WatchDog Driver 2.0
MXC Watchdog # 0 Timer: initial timeout 60 sec
Bluetooth: Virtual HCI driver ver 1.3
Bluetooth: HCI UART driver ver 2.2
Bluetooth: HCIATH protocol initialized
Bluetooth: Generic Bluetooth USB driver ver 0.6
usbcore: registered new interface driver btusb
VPU initialized
mxsdhci: MXC Secure Digital Host Controller Interface driver
mxsdhci: MXC SDHCI Controller Driver.
mmc0: SDHCI detect irq 0 irq 1 INTERNAL DMA
mxsdhci: MXC SDHCI Controller Driver.
mmc1: SDHCI detect irq 203 irq 3 INTERNAL DMA
usbcore: registered new interface driver usbhid
usbhid: USB HID core driver
Cirrus Logic CS42888 ALSA SoC Codec Driver
sgtl5000-i2c 1-000a: SGTL5000 revision 17
No device for DAI imx-ssi-1-0
No device for DAI imx-ssi-1-1
No device for DAI imx-ssi-2-0
No device for DAI imx-ssi-2-1
DMA Sound Buffers Allocated:UseIram=1 buf->addr=f8002000 buf->area=e0842000 size=24576
DMA Sound Buffers Allocated:UseIram=1 buf->addr=cf480000 buf->area=fa310000 size=24576
asoc: SGTL5000 <-> imx-ssi-2-0 mapping ok
mmc0: new SDHC card at address f8a3
mmcblk0: mmc0:f8a3 SU04G 3.69 GiB
 mmcblk0: p1
ALSA device list:
 #0: imx-3stack (SGTL5000)
TCP cubic registered
NET: Registered protocol family 17
can: controller area network core (rev 20090105 abi 8)
NET: Registered protocol family 29
can: raw protocol (rev 20090105)
can: broadcast manager protocol (rev 20090105 t)
Bluetooth: L2CAP ver 2.14
Bluetooth: L2CAP socket layer initialized
Bluetooth: SCO (Voice Link) ver 0.6
Bluetooth: SCO socket layer initialized
Bluetooth: RFCOMM TTY layer initialized
Bluetooth: RFCOMM socket layer initialized
Bluetooth: RFCOMM ver 1.11
Bluetooth: BNEP (Ethernet Emulation) ver 1.3
Bluetooth: BNEP filters: protocol multicast
Bluetooth: HIDP (Human Interface Emulation) ver 1.2
VFP support v0.3: implementor 41 architecture 3 part 30 variant c rev 2
regulator_init_complete: incomplete constraints, leaving DA9052_BUCK_MEM on
regulator_init_complete: incomplete constraints, leaving DA9052_BUCK_PRO on
regulator_init_complete: incomplete constraints, leaving DA9052_BUCK_CORE on
regulator_init_complete: incomplete constraints, leaving DA9052_LDO10 on
regulator_init_complete: incomplete constraints, leaving DA9052_LDO8 on
regulator_init_complete: incomplete constraints, leaving DA9052_LDO7 on
regulator_init_complete: incomplete constraints, leaving DA9052_LDO6 on
regulator_init_complete: incomplete constraints, leaving DA9052_LDO4 on
regulator_init_complete: incomplete constraints, leaving DA9052_LDO3 on
regulator_init_complete: incomplete constraints, leaving DA9052_LDO2 on
regulator_init_complete: incomplete constraints, leaving DA9052_LDO1 on
mxc_rtc mxc_rtc.0: setting system clock to 1970-01-01 00:00:00 UTC (0)
eth0: Freescale FEC PHY driver [Generic PHY] (mii_bus:phy_addr=0:00, irq=-1)
IP-Config: Complete:
    device=eth0, addr=192.168.201.33, mask=255.255.255.0, gw=192.168.201.98,
    host=192.168.201.33, domain=, nis-domain=(none),
    bootserver=192.168.201.98, rootserver=192.168.201.98, rootpath=
Freeing init memory: 156K
Checking rootfs signature Success
Starting logging: OK
Starting mdev...
```

```
Starting portmap: done
Initializing random number generator... read-only file system detected...done
Starting network...
ip: RTNETLINK answers: File exists
Starting Network Interface Plugging Daemon: eth0.
Starting dropbear sshd: OK
Starting stunnel: Reading configuration from file /etc/stunnel/stunnel.conf
PRNG seeded successfully
stunnel.pem: No such file or directory (2)
OK
Starting NFS statd: done
Starting NFS services: exportfs: could not open /var/lib/nfs/.etab.lock for locking: errno 30 (Read-only file system)
exportfs: can't lock /var/lib/nfs/etab for writing
exportfs: could not open /var/lib/nfs/.xtab.lock for locking: errno 30 (Read-only file system)
exportfs: can't lock /var/lib/nfs/xtab for writing
done
Starting NFS daemon: rpc.nfsd: Unable to access /proc/fs/nfsd errno 2 (No such file or directory).
Please try, as root, 'mount -t nfsd nfsd /proc/fs/nfsd' and then restart rpc.nfsd to correct the problem
done
Starting NFS mountd: done
Starting domain name daemon: namedwarning: `named' uses 32-bit capabilities (legacy support in use)
failed
Welcome to Loco
loco login:
```

Category:

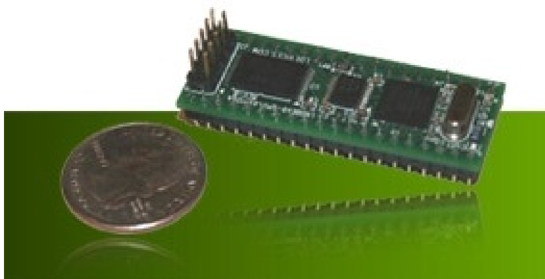
- ARM Development Boards

> From: eLinux.org

Hammer Board

Contents

Hammer Board Specifications



TinCanTools Website

The Hammer CPU board from TinCanTools,is based upon Samsung's S3C2410 ARM920T processor. It gives you the power of an ARM9 processor in an easy to use modular package. The Hammer CPU board fits into a standard 40 pin DIP socket. It is an ideal platform for code development and prototyping in an embedded environment. You can quickly interface to the Hammer using your standard prototyping tools that are based upon 0.1 inch centers.

```
    * 100 MIPS Microprocessor ( CPU ): S3C2410 - Samsung (200 MHz)
    * ARM 920T core with Cache (16K+16K) and MMU
    * Main Memory: 32MB SDRAM (16M x 16 bit, 100MHz)
    * FLASH : 16MB NOR Flash
    * Peripherals available:
          o 2 UART's (also supports IrDA)
          o 1 I2C
          o 2 SPI's
          o 2 16-bit Timers/PWM's
          o 1 8-bit LCD Interface + control signals
          o 1 USB Host Port
          o 1 USB Slave Port
          o 2 ADC's (10 bit )
          o 4 External Interrupt pins
          o 1 Up to 30 pins of GPIO's
    * JTAG Interface: 2 x 7 Header - standard ARM JTAG interface
    * Size: 0.75 inches (width) X 2.25 inches (length)
    * Package: Fits a standard 40-pin DIP socket (0.1 inch lead spacing)
    * Power Requirements: +5VDC @ 120 mA (typical)
```

Interface Options

I want to use the Hammer with:

- Ethernet
- BlueTooth
- Zigbee
- 802.11b/g
- WebCam

Resources

- HOWTOs and Tutorials
- Software
- Hardware
- Application Notes

Hammer Details

The Hammer comes preloaded with an open source bootloader, Linux 2.6 kernel, and a uClibc/Busybox based root file system. Board schematics, application notes, and source code are also included on the CD. The Hammer Board is a 40 pin DIP modules that can run full standard GNU/Linux.

The Hammer Kit comes with everything you will need to begin developing for the Hammer Board, and is **strongly recommended** for first time Hammer Board users. All development tools have been created for use with Linux, so you will need a working Linux system to use the Hammer Board and develop applications for it.



Hammer Board Pinouts

Pin Function	Alt. Function	40-Pin DIP		Pin Function	Alt. Function
RXD0	GPH3	1	40	+5.0 VDC INPUT	
TXD0	GPH2	2	39	+3.3VDC OUTPUT	
IICSDA	GPE15	3	38	DN0	USB HOST (-)
IICSCL	GPE14	4	37	DP0	USB HOST (+)
SPIMISO0	GPE11	5	36	DN1	USB SLAVE (-)
SPIMOSI0	GPE12	6	35	DP1	USB SLAVE (+)
SPICLK0	GPE13	7	34	AIN0 (ADC INPUT - CH 0)	
nSS0	EINT10	8	33	AIN1 (ADC INPUT - CH 1	
nRESET	nRESET	9	32	EINT13	SPIMISO1
nRTS0	GPH1	10	31	EINT14	SPIMOSI1
nCTS0	GPH0	11	30	EINT15	SPICLK1
TOUT0	GPB0	12	29	EINT11	nSS1
TOUT2	GPB2	13	28	LCD_VD7	GPC15
RXD2	GPH7	14	27	LCD_VD6	GPC14
TXD2	GPH6	15	26	LCD_VD5	GPC13
VM	GPC4	16	25	LCD_VD4	GPC12
VFRAME	GPC3	17	24	LCD_VD3	GPC11
VLINE	GPC2	18	23	LCD_VD2	GPC10
VCLK	GPC1	19	22	LCD_VD1	GPC9
GND	GND	20	21	LCD_VD0	GPC8

LED

Pinout with more explanations and links.

Pinout with more explanations and links.

Pin number	Main Function	Note	Alt Function	Note
1	RXD0	RS232 Receive	GPH3	GPIO port H bit 3
2	TXD0	RS232 Transmit	GPH2	GPIO port H bit 2
3	IICSDA	I2C SDA Wikipedia explains I2C	GPE15	GPIO port E bit 15
4	IICSCL	I2C SCL	GPE14	GPIO port E bit 14
5	SPIMISO0	SPI port 0 MISO Wikipedia explains SPI	GPE11	GPIO port E bit 11
6	SPIMOSI0	SPI port 0 MOSI	GPE12	GPIO port E bit 12
7	SPICLK0	SPI port 0 clock	GPE13	GPIO port E bit 13
8	nSS0	SPI port 0 select	EINT10	External Interrupt 10 (Wakeup capable?)
9	nRESET	CPU reset.		
10	nRTS0	RS232 RTS (Ready to send)	GPH1	GPIO port H bit 1
11	nCTS0	RS232 CTS (Clear to send)	GPH0	GPIO port H bit 0
12	TOUT0	PWM timer 0 out	GPB0	GPIO port B bit 0
13	TOUT2	PWM timer 2 out	GPB2	GPIO port B bit 2
14	RXD2	RS232 port 2 receive (Note, Port 1 isn't accessible)	GPH7	GPIO port H bit 7
15	TXD2	RS232 port 2 transmit	GPH6	GPIO port H bit 6
16	VM	LCD AC bias	GPC4	GPIO port C bit 4
17	VFRAME	LCD Vertical sync	GPC3	GPIO port C bit 3

18	VLINE	LCD Horizontal sync	GPC2	GPIO port C bit 2
19	VCLK	LCD pixel clock	GPC1	GPIO port C bit 1
20	GND	Ground - supply voltage return.		
21	LCD_VD0	LCD data bit 0	GPC8	GPIO port C bit 8
22	LCD_VD1	LCD data bit 1	GPC9	GPIO port C bit 9
23	LCD_VD2	LCD data bit 2	GPC10	GPIO port C bit 10
24	LCD_VD3	LCD data bit 3	GPC11	GPIO port C bit 11
25	LCD_VD4	LCD data bit 4	GPC12	GPIO port C bit 12
26	LCD_VD5	LCD data bit 5	GPC13	GPIO port C bit 13
27	LCD_VD6	LCD data bit 6	GPC14	GPIO port C bit 14
28	LCD_VD7	LCD data bit 7	GPC15	GPIO port C bit 15
29	EINT11	External interrupt 11	nSS 1	SPI port 1 select
30	EINT15	External interrupt 15	SPICLK1	SPI port 1 Clock
31	EINT14	External interrupt 14	SPIMOSI1	SPI port 1 MOSI
32	EINT13	External interrupt 13	SPIMISO1	SPI port 1 MISO
33	AIN1	ADC input 1		(Note - these ADCs are independant, and do not support differential mode)
34	AIN0	ADC input 0		
35	DP1	USB 1 Slave +		
36	DN1	USB 1 Slave -		
37	DP0	USB 0 Host +		
38	DN0	USB 0 Host -		
39	+3.3VDC	Internal regulator output. Can supply 100mA(?) safely. Overloading this may cause irreparable damage.		
40	+5V	5V input.		

Related Products

Hammer Carrier Board - the original carrier board for Hammer

Nail Kit - Great development system for Hammer to take where ever you go.

Hammer-RDP - the Robotics Development Platform

Hammer Project pages

Christmas Lights Controller Audio Mixer Project

Categories:

- TCT-Hammer
- TinCanTools

- ARM Development Boards

> From: eLinux.org

Hawkboard

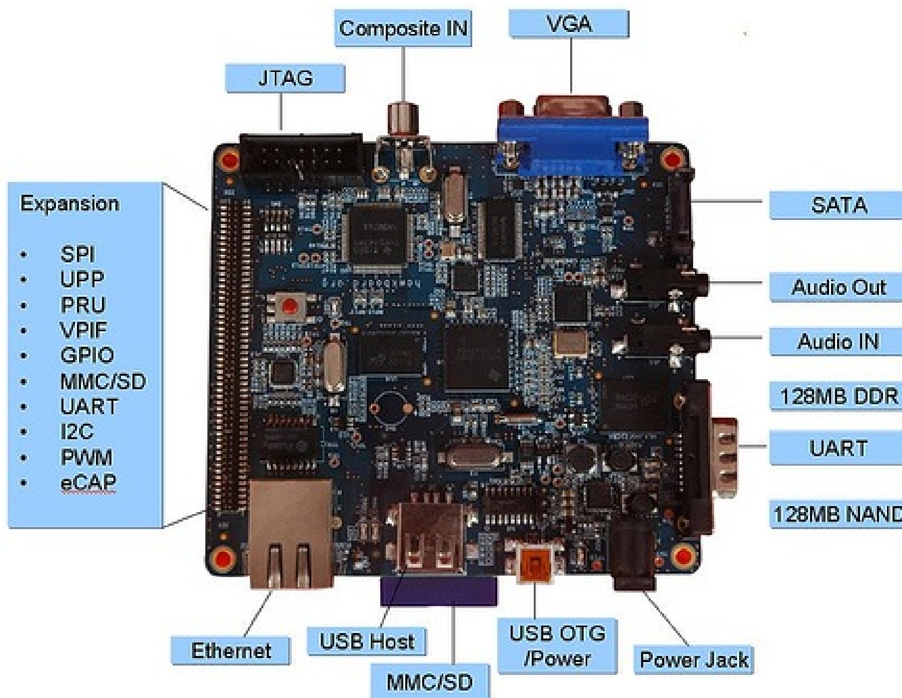


Access Beginners Guide to Hawkboard if you are new to Development on Hawkboard
http://elinux.org/Hawkboard/BeginnersGuide

Contents

- 13 Subpages

Hardware Details



- **HawkBoard User Manual**
- **HawkBoard Schematics**
- Technical Reference Manual for OMAP L 138 Processor is Here
- Applications of OMAP L 138 are Here

Software Details

Tools & Softwares Downloads

- **ARM Cross Compiler - For Linux/U-boot**

Note:

1.Systems running "full" Linux, i.e., Linux on CPUs with an MMU. Use this to build both the Linux kernel and applications.

2.New Releases : http://www.codesourcery.com/sgpp/lite/arm/portal/release1039

- **DSP Cross Compiler -TI's Code Generation Tool (CGT) c674x via option -mv6740**
- **TI CCS 4 Plantinum**
- **ARM Cross Compiler - Bare Metal**

Note:

1.This is for RTOS systems or "bare metal" systems where no operating system is present. **These toolchains should not be used to build Linux kernels or Linux applications**.

2.New Releases : http://www.codesourcery.com/sgpp/lite/arm/portal/release1033

- **AIS Generator / UART Host Tool (Needs .NET Framework)**

Tools From Other Vendors

- ARM Cross Compiler provided by impactlinux
- ARM Native Compiler (to be used directly on Hawkboard) provided by impactlinux
- ARM Cross Compiler provided by Ridgerun
- Fedora Cross Compiler

JTAG emulators

- Application Note for using CCSv4 & XDS Emulator with Hawkboard

Misc Tools

- VirtualBox -To run Linux/Windows simultaneously with Windows/Linux
- Tftp Server
- Teraterm-Tool to Access Serial Console(Alternative to Hyperterminal) + SSH Client

Sources

- Linux Kernel
- u-boot

Sample Rootfs

1.Ubuntu Jaunty http://hawktool.googlecode.com/files/RootFS_v1.tar.bz2 (username/password =hawk/password)

2.Fedora RootFS -http://ftp.linux.org.uk/pub/linux/arm/fedora/rootfs/rootfs-f12.tar.bz2 (username/password =root/fedoraarm)

3.Impactlinux http://impactlinux.com/fwl/downloads/binaries/root-filesystem-armv5l.tar.bz2

Getting Started

1.**Powering the Device**: Hawkboard can be powered through USB OTG port(mini USB) or Separate 5V Source. Since the USB port of a PC/laptop or hub are often limited to 500 mA, it is advisable to use a separate power supply of 5VDC that supplies at least 1 A of current.

Note: It is recommended to use a separate 5VDC Power supply with at least a 1 Amps current rating instead of USB Power for Normal Operation of the Board

2.**Connecting VGA Monitor**: Once the power supply has been attached, Hawkboard boots to Uboot Bootloader and will show Hawkboard Logo on Screen.

3.**Connecting UART**: To transfer images to Hawkboard and to set other parameters, the UART/serial cable needs to be attached to Hawkboard and PC. A NULL Modem(Crossed Cable i.e 2 and 3 Crossed) is needed for that. Run a terminal session (such as Minicom on Linux or TeraTerm on Windows) on the Host PC and configure it to connect to that serial port with the following characteristics:

Bits per Second: 115200

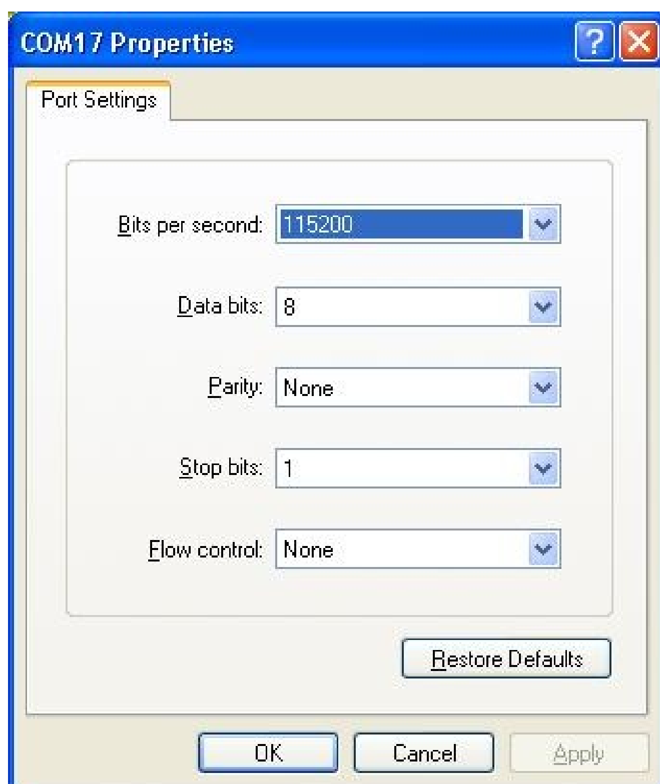
Data Bits: 8

Parity: None

Stop Bits: 1

Flow Control: None

Transmit delay: 0 msec/char, 10 msec/line



After powering the device you should be able to see the following:

```
U-Boot 2009.01-dirty (Nov 18 2009 - 23:30:48)

DRAM:  128 MB
NAND:  NAND device: Manufacturer ID: 0x2c, Chip ID: 0xa1 (Micron NAND 128MiB 1,8
V 8-bit)
Bad block table found at page 65472, version 0x01
Bad block table found at page 65408, version 0x01
128 MiB
*** Warning - bad CRC or NAND, using default environment

In:    serial
Out:   serial
Err:   serial
ARM Clock : 300000000 Hz
DDR Clock : 150000000 Hz
Ethernet PHY: GENERIC @ 0x07
Hit any key to stop autoboot:  0
hawkboard.org >
```



- On Hyperterminal (Any other Serial Console) (Left) On VGA Monitor you will see HawkBoard Logo (Right)

Command prompts in this guide

In this guide, commands are preceded by prompts that indicate the environment where the command is to be typed. For example:

- **host$**

Indicates command to be typed into the shell window of the host Linux workstation.

- **Hawkboard.org>**

Indicates commands to be typed into the U-Boot shell in a console window connected to the Hawkboard's serial port.

- **target$**

Indicates commands to be typed into the Linux shell in the terminal window connected to the Hawkboard's serial port or TTY..

Building & Booting Uboot images from source

HawkBoard comes with pre-installed U-boot and normally you don't want to Compile and Load U-boot unless you have Bricked the Bootloader or want to change something.So You can Skip this Section and can go Directly to Compiling Linux Kernel or Booting Kernel Section.

Setting up the Linux Environment

Most of the following instruction for UART booting are meant for Windows environment. For people who only have Linux on their machines the UART booting steps and creating the UART and NAND image from the ELF files seems impossible. But nothing is impossible on Linux. Hence the way out. The following instructions are Fedora specific. If you have a Debian based distro please make the corresponding changes with apt-get.

- Installing WINE and MONO

```
host# yum install wine mono* -y
```

You need to install wine and mono(make sure your mono is v2.4 and above) packages. They will help you install and run the TI provided development tools.

- Installing the TI development tools

```
host$ wine AISgen_d800k002_Install_v1.3.exe
```

These tools are installed in your wine/drive_C directory where-ever that is defined for your particular distribution. For Fedora the steps are as follows, from your home directory.

- Using the actual tools

```
host$ cd .wine/drive_c/Program Files/Texas Instruments/AISgen for D800K002/
```

- Running the AISgen tool

```
host$ mono AISgen_d800k002.exe
```

- Running the UartHost tool

```
host$ cd UartHost
host$ mono UartHost.exe
```

- Setting up your COMPORT

```
host$ cd .wine/dosdevices
host# ln -s /dev/ttyS1 com1
```

Now you can follow all the instructions given below from your Linux machine.

Compiling u-boot (bootloader)

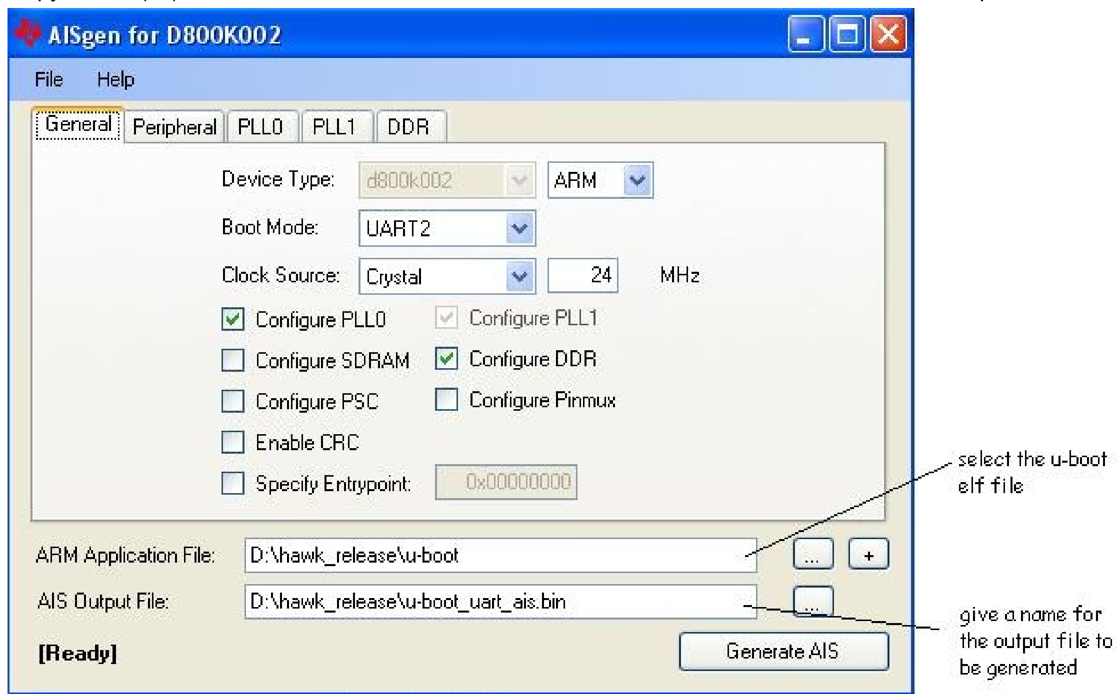
1. Issue compile commands with make:

```
host$ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- distclean
host$ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- omapl_hawkboard_config
host$ make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi-
```

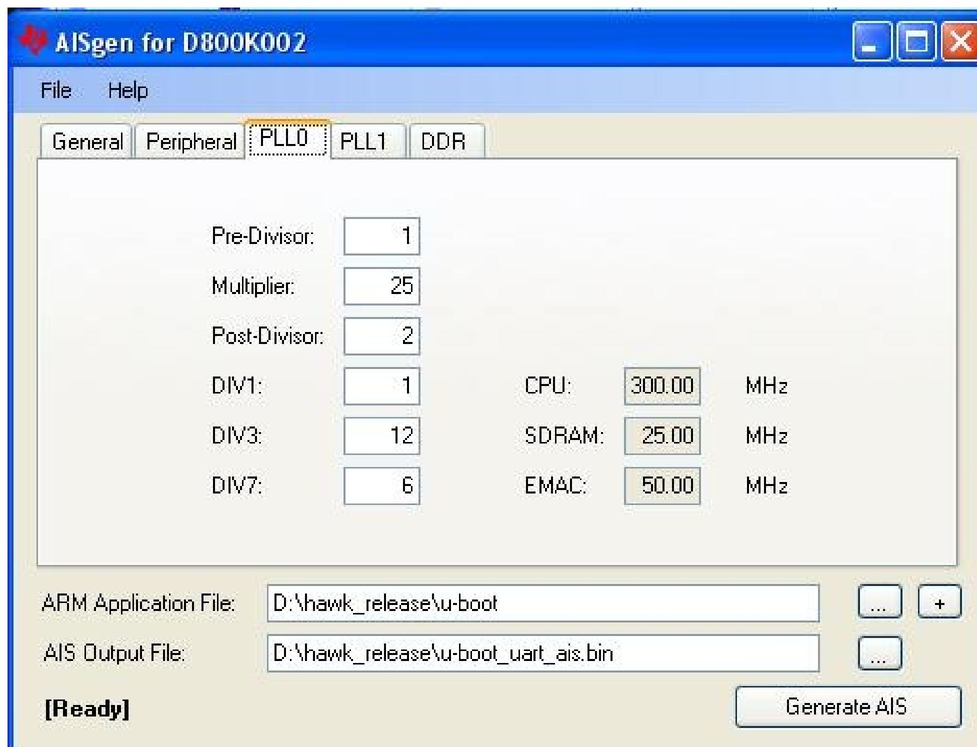
If you get an error while compiling, see the FAQor the mailing list thread

Signing u-boot for UART boot

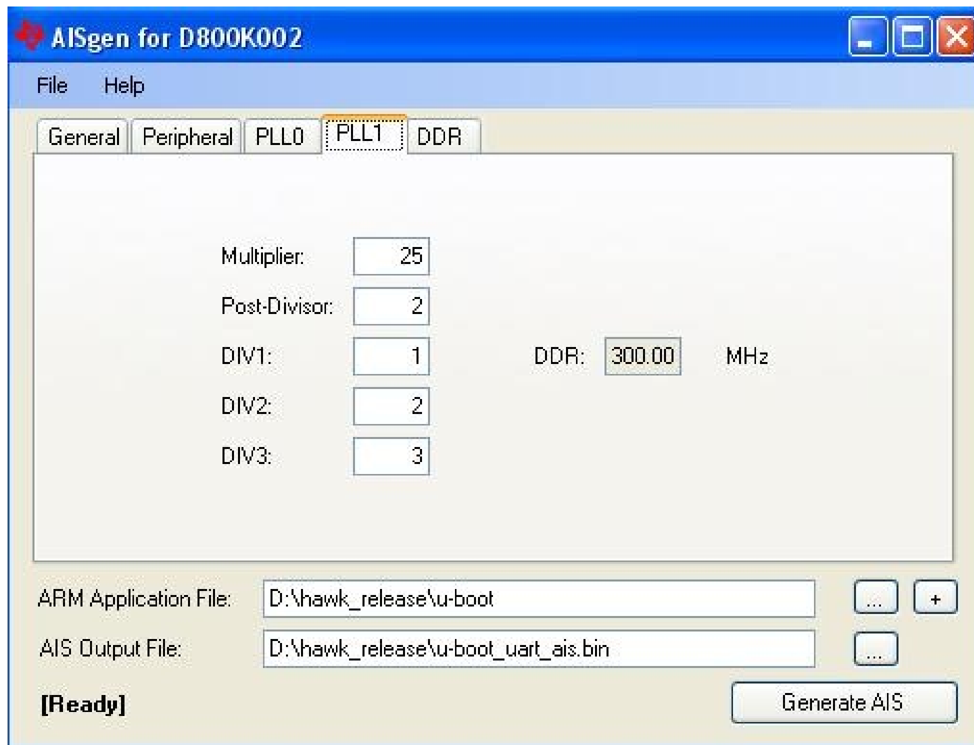
1. Copy u-boot (elf) file to the host machine where AIS Generator is installed. Follow the below steps

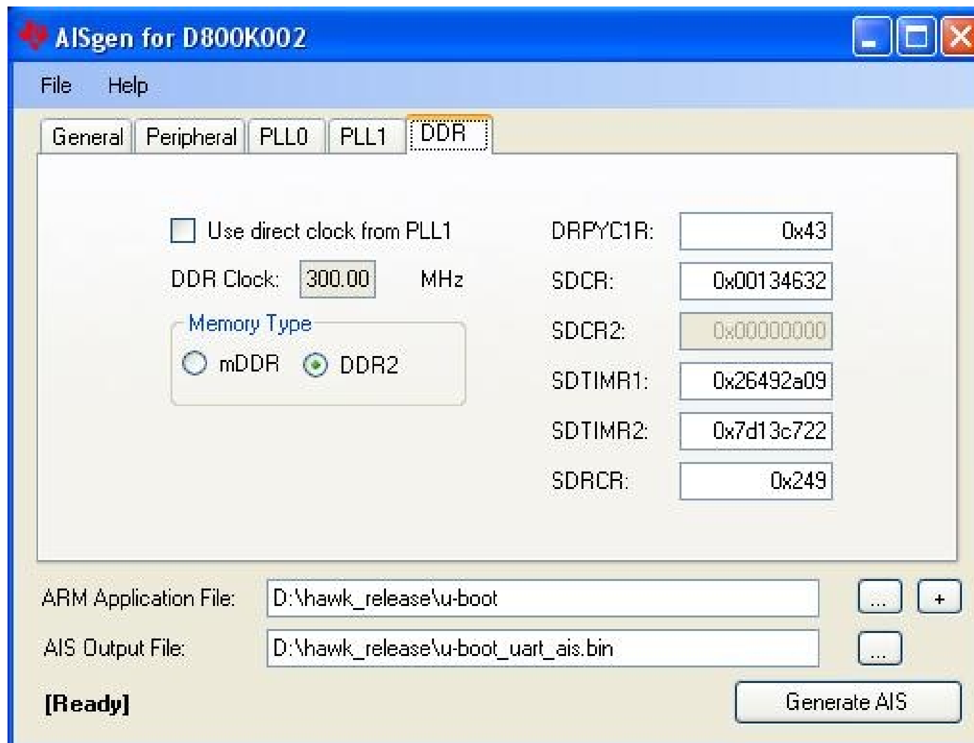


2.



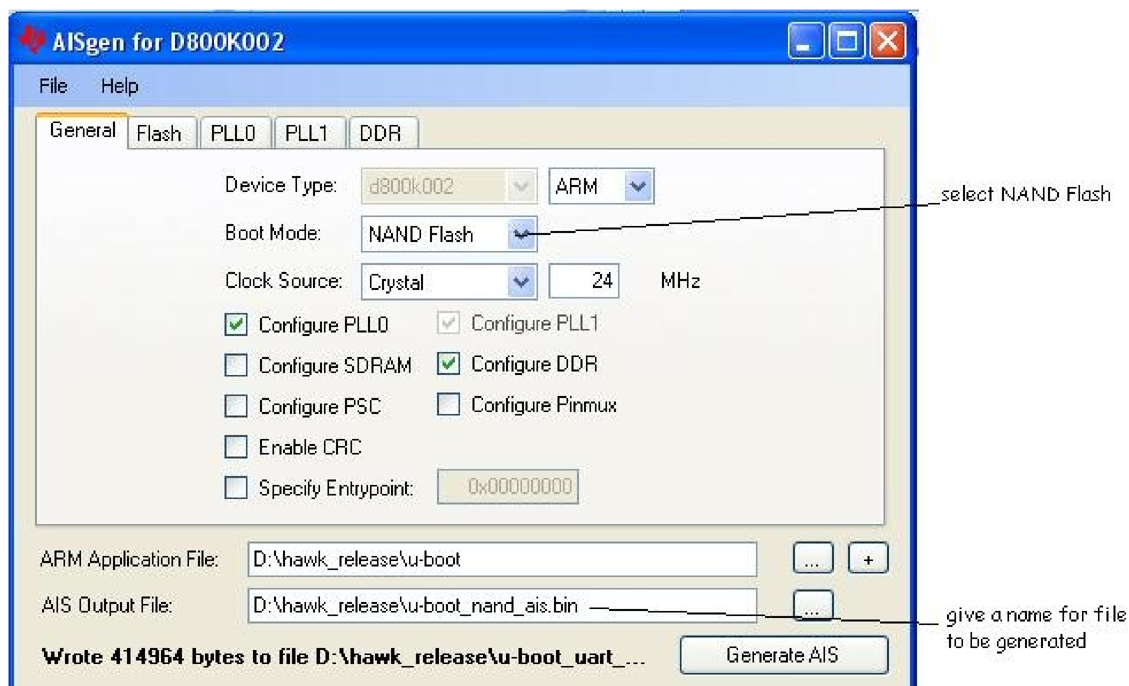
3.

4. 

5. 

Signing u-boot for NAND boot

1. Copy u-boot (elf) file to the host machine where AIS Generator is installed. Follow the below steps

2. 

3. Configure PLL0, PLL1, DDR tabs as shown above. The File name to generate remains as in the step above.

Booting

Booting u-boot over UART

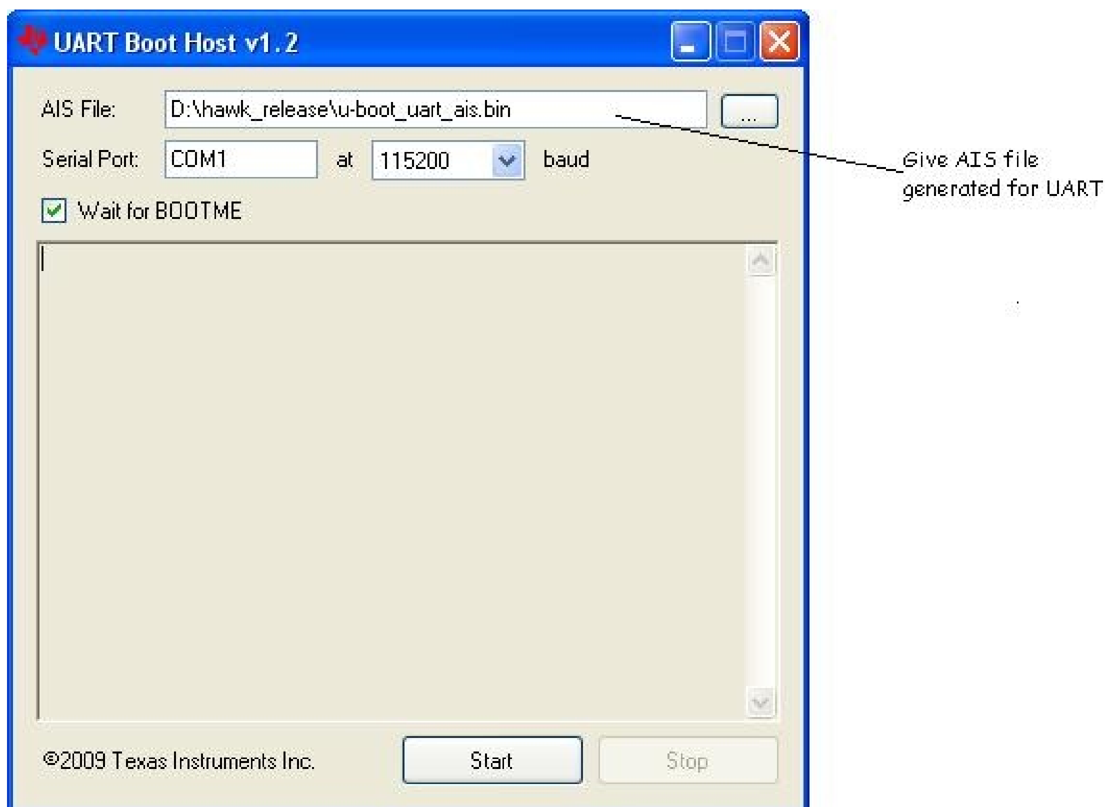
- Power OFF the board
- Close any previously running terminal programs
- Connect the UART cable(Null/Cross Cable) to the Host machine.

Following process used on Windows family of Operating System:

- User can use mono utility with dvflasher.ext program to load u-boot on Linux system.
- Details To be updated

Following process used on Windows family of Operating System:

- Configure the Boot Switches as 1-OFF 2-ON 3-OFF 4-ON
- Start the UART Host Utility, should have been installed with AIS Generator / UART Host Tool
- Select the AIS FILE Generated for UART



- Click on the Start button
- Power UP the Device
- If you see errors click on stop and press start again and give a board reset.
- Wait till you all the below messages and COM is closed:

```
(File IO): Read 414964 bytes from file D:\hawk_release\u-boot_uart_ais.bin.
(Serial Port): Opening COM1 at 115200 baud...
(AIS Parse): Read magic word 0x41504954.
(AIS Parse): Waiting for BOOTME...
(AIS Parse): Performing Start-Word Sync...
(AIS Parse): Performing Ping Opcode Sync...
(AIS Parse): Processing command 0: 0x5853590D.
(AIS Parse): Performing Opcode Sync...
(AIS Parse): Executing function...
(AIS Parse): Processing command 1: 0x5853590D.
(AIS Parse): Loaded 1512-byte section to address 0xC10E4BEC.
...
...
...
(AIS Parse): Processing command 15: 0x58535906.
(AIS Parse): Performing Opcode Sync...
(AIS Parse): Performing jump and close...
(AIS Parse): AIS complete. Jump to address 0xC1080000.
(AIS Parse): Waiting for DONE...
(AIS Parse): Boot completed successfully.
(Serial Port): Closing COM1.
```

- Now Start any standard UART Terminal and hit enter key, should see the u-boot prompt

Procedure to flash u-boot on NAND

After booting the u-boot over UART as mentioned above,

- On the u-boot prompt in the terminal window
- Configure the Ethernet server and Client IP addresses, For e.g.

```
hawkboard.org > setenv serverip 172.24.156.199
hawkboard.org > setenv ipaddr 172.24.190.58
```

- Download the u-boot generated for NAND

```
hawkboard.org > tftpboot 0xc0700000 u-boot_nand_ais.bin

TFTP from server 172.24.156.199; our IP address is 172.24.190.58
Filename 'u-boot_nand_ais.bin'.
Load address: 0xc0700000
Loading: #################################################################
         #################
done
Bytes transferred = 414988 (6550c hex)
```

- Erase NAND Flash

```
hawkboard.org > nand erase

NAND erase: device 0 whole chip
OK
```

- Flash the NAND with u-boot

```
hawkboard.org > nand write.e 0xc0700000 0x20000 0x70000

NAND write: device 0 offset 0x20000, size 0x70000
458752 bytes written: OK
hawkboard.org >
```

- Switch off the board
- Change the DIP Switches for NAND boot 1-ON 2-OFF 3-OFF 4-OFF

Booting u-boot over NAND

Flash u-boot into NAND as mentioned above

- Switch off the board
- Change the DIP Switches for NAND boot 1-ON 2-OFF 3-OFF 4-OFF
- Switch on the board

Building & Booting Kernel images from source

Compiling Linux Kernel

1. Issue compile commands with make:

```
make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- distclean
make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- omapl138_hawkboard_defconfig
make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabi- uImage
```

If you get the *"mkimage" command not found - U-Boot images will not be built* error,build uboot from the source first and then copy the mkimage binary from ~u-boot-omapl1/tools to a directory in your PATH

Booting

Booting Linux Kernel and Mounting RAMDISK

You should find a kernel and a ramdisk image if you haven't found already. You can download samples from [1] (uImage_v1 & ramdisk_v1.gz)). Then copy these files to your "/tftpboot" directory assuming you will use tftp for transferring files to your board. In order to do so you need to run a tftp server on your system. Under linux a good choice is atfptd, but you can also use others. Typically a linux tftp server exports the directory /tftpboot. Make sure before you continue that your ftp server is running. At the u-boot prompt:

Set ethernet connection:

```
$setenv serverip <PC ipaddress>
$setenv ipaddr <board ipaddress>
```

\ should be the address of the PC running the tftp server. \ should be an address that is not in use on the same subnet as the tftp server. Transfer files to the board:

```
$tftp c0700000 uImage_v1
$tftp c1180000 ramdisk_v1.gz
```

If you broke your bootargs previously then:

```
$setenv bootargs "mem=128M console=ttyS2,115200n8 root=/dev/ram0 rw initrd=0xc1180000,4M"
```

And finally boot the images:

```
$bootm c0700000
```

The TFTP transferred images can be written to the NAND Flash for future use to perform kernel upgrades and enable booting without network connection.

To do this erase the sections of the NAND containing the kernel and file system and write the images from RAM to the NAND.

```
$nand erase 200000 200000
$nand write.e 0xc0700000 0x200000 0x200000

$nand erase 0x400000 0x300000
$nand write.e 0xc1180000 0x400000 0x300000
```

The written images can be read back using

```
$nand read.e 0xc0700000 0x200000 0x200000
$nand read.e 0xc1180000 0x400000 0x300000
```

Set the bootcmd to read these images from NAND automatically and boot:

```
$setenv bootcmd 'nand read.e 0xc0700000 0x200000 0x200000;nand read.e 0xc1180000 0x400000 0x300000;bootm 0xc0700000'
```

Booting Linux Kernel and Mounting NFS (Network File System)

Load the Kernel through NAND Flash or TFTP Method ,Once kernel has been loaded to RAM using tftp c0700000 uImage_v1 or similar Next. Follow following steps to Mount RootFS through NFS.(Ubuntu)

Setting NFS Server

1. Install NFS Server

   host$ sudo apt-get install portmap nfs-kernel-server

2. Share the Folder containing extracted RootFS (e.g /nfsroot contains RootFS)

Edit /etc/exports and add the shares:

```
/nfsroot <NETWROK_IP><SUBNET_MASK>(rw,sync,no_subtree_check,no_root_squash)
```

e.g, /nfsroot 192.168.0.0/255.255.255.0(rw,sync,no_subtree_check,no_root_squash)

Assuming your network is is 192.168.0.0

NB:- no_root_squash might be necessary if you are getting sudo erros in Ubuntu Rootfs

3.After setting up /etc/exports, export the shares:

```
 host$ sudo exportfs -ra
```

restart the nfs-server if required

```
 host$ sudo /etc/init.d/nfs-kernel-server restart
```

Booting Board using NFS

Under Uboot prompt once the Kernel has been loaded through

```
 hawkboard.org> tftp c0700000 uImage_v1
```

1.Set bootargs

Set boot arguments:

a)If Board uses Static IP

```
hawkboard.org> setenv bootargs 'console=ttyS2,115200n8 noinitrd rw ip=<HawkboardIP>:<server-ip>:<gateway-ip>:<netmask
>::eth0: root=/dev/nfs nfsroot=<NFS Server ipaddress>:/nfsroot'
```

e.g

```
 hawkboard.org> setenv bootargs 'console=ttyS2,115200n8 noinitrd rw ip=192.168.0.125:192.168.0.1:192.168.0.1:255.255.
255.0::eth0: root=/dev/nfs nfsroot=<NFS Server ipaddress>:/nfsroot'
```

where 192.168.0.125 is board IP & 192.168.0.1 is gateway IP& Router IP.

b)DHCP IP Board

```
 hawkboard.org> setenv bootargs "mem=128M console=ttyS2,115200n8 root=/dev/nfs nfsroot=<Server ipaddress>:/nfsroot ip
=dhcp"
```

2.And finally boot the image:

```
 hawkboard.org> bootm c0700000
```

Booting Linux Kernel from USB

This steps work for USB Pendrive connected to Hub or Directly to USB Standard Port and rootfs is Ubuntu jaunty made through rootstock.Should work for other distribution too.

1. Load uimage through tftp (as usual)

setenv serverip setenv ipaddr tftp c0700000 uImage_v1

2.Sent environment variable

```
setenv bootargs console=ttyS2,115200n8 noinitrd root=/dev/sda1 rootwait rw init=/sbin/init
```

3.bootm

```
bootm c0700000
```

Here /dev/sda1 is USB Drive with only one ext2 partition and rootfs lying init.

Booting Linux Kernel from SATA

Currently the provided u-boot does not allow booting from either SATA. However, it is possible to boot from nand and have the root filesystem loaded from SATA. If you want to do so, you have to figure out the right partition for the root filesystem and add that to your bootargs.

E.g. in u-boot say something like:

```
setenv bootargs mem=128M console=ttyS2,115200n8 root=/dev/sda1 rootwait
```

Where you replace /dev/sda1 with the name of the device that contains your root filesystem. /dev/sda1 is a good name for sata partition 1 if no usb or sd devices are present. Note that the rootwait argument is needed. It tells the kernel to wait until the disk has settled.

Booting Linux Kernel from SD OR MMC

Currently the provided u-boot does not allow direct booting from either SD or MMC However, it is possible to boot from nand and have the root filesystem loaded from SD If you want to do so, you have to figure out the right partition for the root filesystem and add that to your bootargs.

E.g. in u-boot say something like:

```
setenv bootargs console=ttyS2,115200n8 console=tty1 noinitrd root=/dev/mmcblk0p1 rootwait rw
```

or

```
setenv bootargs console=ttyS2,115200n8 console=tty1 noinitrd root=/dev/mmcblk0p1 rootdelay = 2 rootfstype = ext2 rw
```

Here p1 in mmcblk0p1 says that Rootfs is present in first partition of SD Card.

MISC

1.If you find screen broken, do this before bootm in u-boot, ideally this should go into kernel code

```
mw.l 0x01c14110 0x44442222 1;mw.l 0x01c14114 0x44400000 1;mw.l 0x01c14118 0x04604404 1;
```

2.For booting android over MMC the bootargs should be

```
setenv bootargs mem=128M console=ttyS2,115200n8 noinitrd root=/dev/mmcblk0p1 rootwait ip=off init=/init androidboot.c
onsole=ttyS2
```

3.You probably want to avoid that you have to retype these commands every time you want to boot. This can be achieved easily by issuing the following commands on the u-boot prompt:

```
setenv serverip <PC ipaddress>
setenv ipaddr <board ipaddress>
setenv bootargs_nfs mem=128M console=ttyS2,115200n8 root=/dev/nfs nfsroot=<PC ipaddress>:/nfsroot ip=dhcp
setenv bootcmd 'setenv bootargs $bootargs_nfs;tftp c0700000 uImage.v1; bootm c0700000'
saveenv
```

Make sure to use single quotes in the last setenv command.

Custom RootFS

1.Fedora http://fedoraproject.org/wiki/Architectures/ARM/RfsBuild

2.http://arago-project.org

3.Ubuntu Use Rootstock

4.Angstrom uses Openembedded http://www.angstrom-distribution.org/narcissus/

Making Use of the C6740 DSP Core

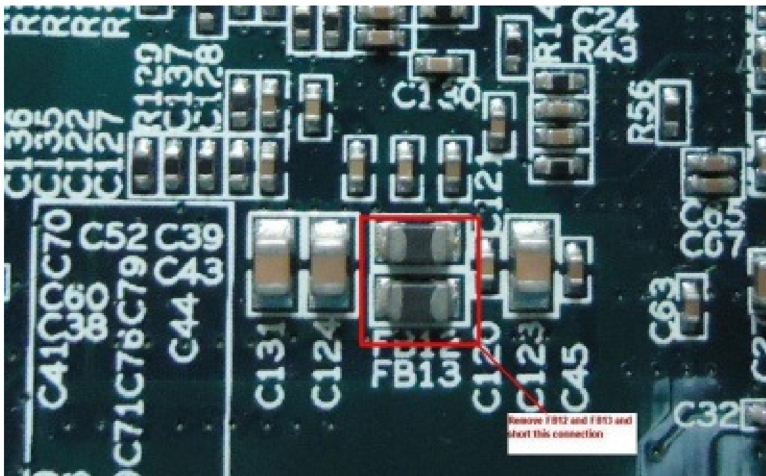
The C6740 DSP core inside the OMAP-L138 processor is extremely capable, providing fixed-point and full double-precision floating-point operations. Probably the simplest way to get started with the C6740 DSP core on the Hawkboard is the C6Run project. It provides a GCC-like front end for building applications, which, when executed from the Linux command prompt, run transparently on the DSP. See the C6Run main page on the TI Embedded Processor wiki for more details and to get started.

FAQs

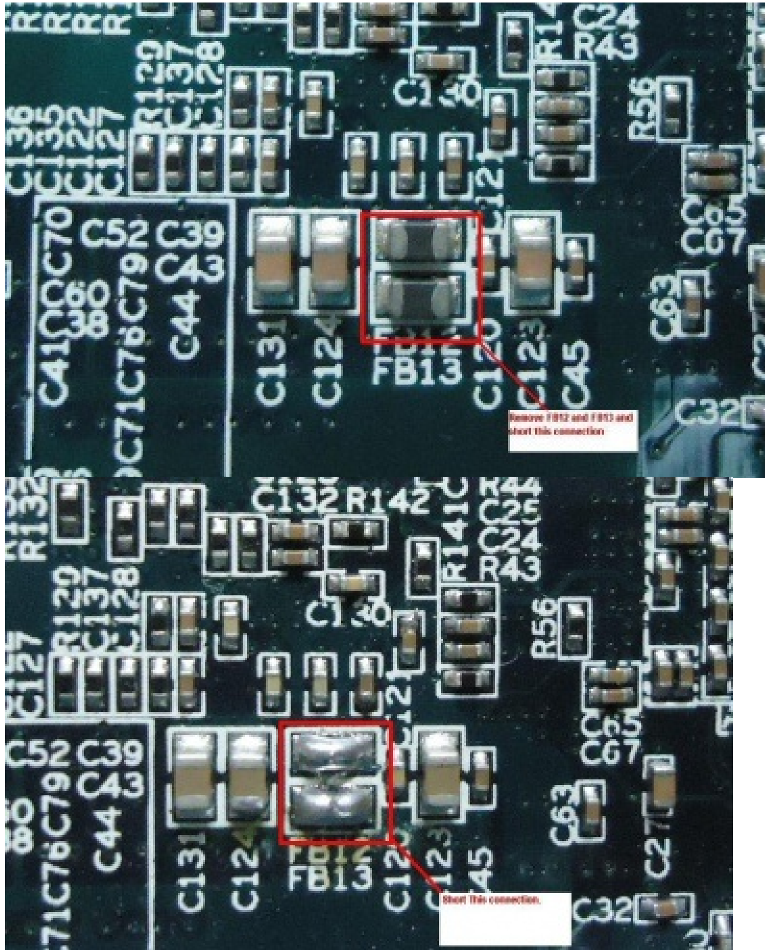
For HawkBoard frequently asked questions (FAQ) see community FAQ.

Common Issues

At this stage it appears that the latest (late 2010) batch of boards have serious problems many problems have been discussed on the mailing list regarding failure to boot linux and problems with tftp There has been a hardware fix posted on www.innovatesolutions.net and discussed HERE. Summary of the fix is either to send the faulty board to distributor who will send them on to innovate alternatively, the fix can also be done by customers without voiding guarantees. The fix is made up by removing two ferrite beads and shortening the pads afterwards.



At this stage it appears that the latest (late 2010) batch of boards have serious problems many problems have been discussed on the mailing list regarding failure to boot linux and problems with tftp There has been a hardware fix posted on www.innovatesolutions.net and discussed HERE. Summary of the fix is either to send the faulty board to distributor who will send them on to innovate alternatively, the fix can also be done by customers without voiding guarantees. The fix is made up by removing two ferrite beads and shortening the pads afterwards.



Projects Based on HawkBoard

Live Links

- Blog : http://hawkboard.wordpress.com/
- Portal : http://hawkboard.org
- Join us : hawkboard on google groups
- IRC : “#hawkboard” on Freenode
- IRC logs : http://ibot.rikers.org/%23hawkboard/
- Twitter : http://twitter.com/hawkboard
- Wikipedia : http://en.wikipedia.org/wiki/Hawk_Board
- Photos(Flickr): http://www.flickr.com/photos/hawkboard
- Mails : hawkboard@googlegroups.com
- Software : code.google.com/p/hawkboard
- OMAP L 138 : http://focus.ti.com/docs/prod/folders/print/omap-l138.html
- Applications : http://wiki.davincidsp.com/index.php/C674x/OMAPL1x_Introductory_Information
- More Details : http://wiki.davincidsp.com/index.php/Category:OMAPL1

Guide Links

> From: eLinux.org

ITSY

The Itsy pocket computer is a flexible research platform whose primary objective is to enable hardware and software research in pocket computing, including low-power hardware, power management, operating systems, wireless networking, user interfaces, and applications.

Itsy's CPU is a StrongARM SA-1100 processor (developed by Digital Equipment Corporation's Digital Semiconductor division, which became part of Intel in May 1998). Itsy contains 16 megabytes of DRAM and 4 megabytes of Flash memory. The main crystal frequency is 3.6864 MHz which, using the processor's phase-locked loop (PLL), results in a variable CPU core frequency from 59.0 MHz to 206.4 MHz.

Article on the ITSY

Interview with Jim Getty About the ITSY

Categories:

- ARM Development Boards
- Historical Development

> From: eLinux.org

LART Project

The LART is a small yet powerful embedded computer capable of running Linux. Its performance is around 250 MIPS while consuming less than 1 Watt of power. In a standard configuration it holds 32MB DRAM and 4MB Flash ROM, which is sufficient for a Linux kernel and a sizeable ramdisk image.

Design summary

In 1998, the researchers of the MMC project found they had a need for a small, powerful computer board that could be used in experiments with wireless multimedia. The board would have to be low-power and inexpensive, as the project would need several of them. As no off-the-shelf solution that offered an acceptable compromise could be found, a new design was made. What's on the mainboard?

Here are the LART mainboard specs in short:

```
    * 220 MHz Digital SA-1100 StrongARM CPU
    * 32 Mbyte EDO RAM
    * 4 MB Intel Fast boot block Flash memory
    * Power usage < 1 W
    * Performance > 200 MIPS
```

The board can run standalone, booting an OS from Flash. The 4 MB Flash is sufficient for a bootloader, a compressed kernel and a compressed ramdisk. The LART accepts an input voltage between 3.5 and 16 V; the on-board DC-DC converters have an efficiency between 90 and 95%.

LART Project Page

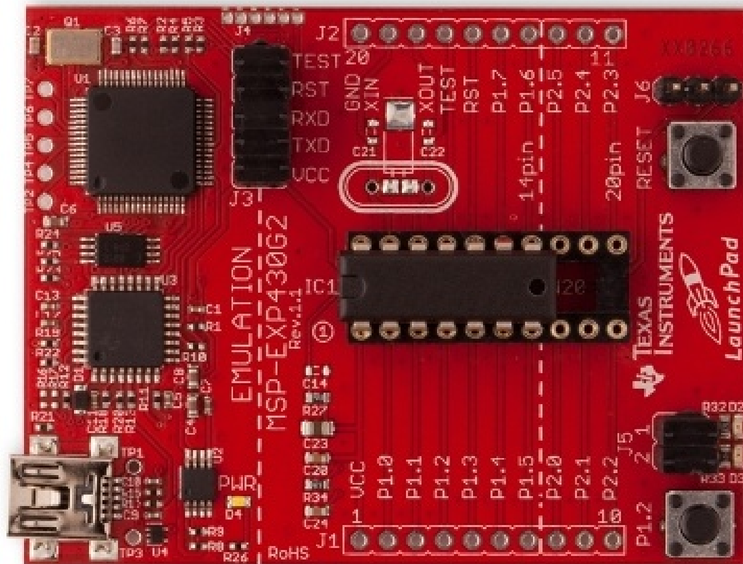
Categories:

- ARM Development Boards
- Historical Development

> From: eLinux.org

Launchpad

A placeholder page for TI's Launchpad development kit.



Links

HOWTO: Launchpad Programming With Linux

Category:

- Development Boards

> From: eLinux.org

LeopardBoard

The Leopard Board is a full featured, ultra low cost, small form factor, high performance development system which includes TMS320DM355 Processor board and a VGA camera board to provide VGA resolution video capture. The board is manufactured by Leopard Imaging and is available at [1]. A community website gives access to documentation and community forum leopardboard.org

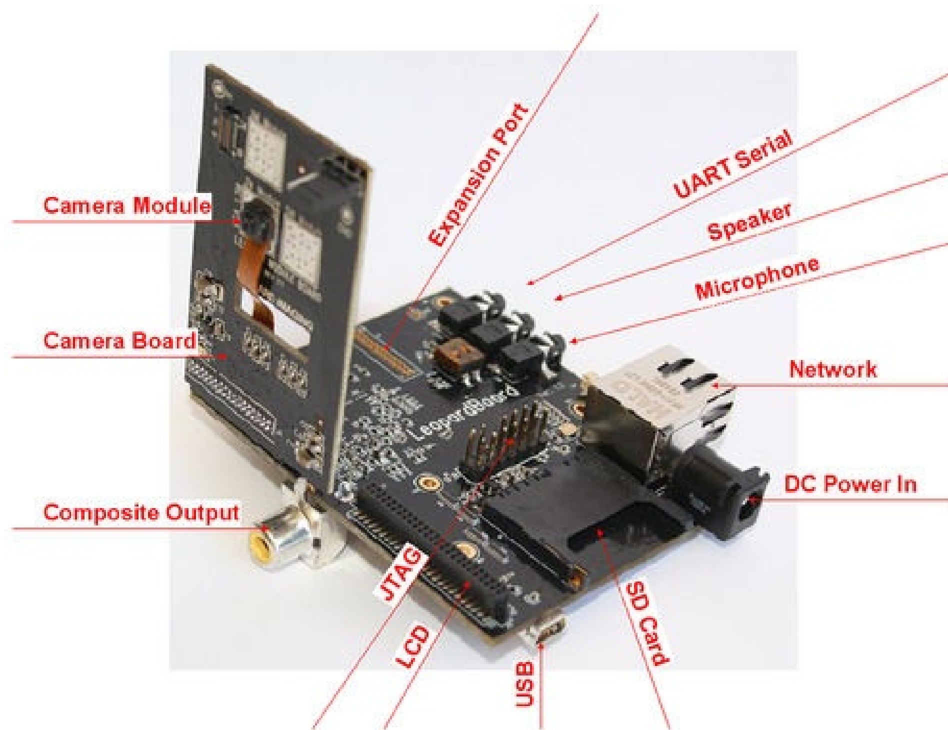
Contents

Helpful Links

- LeopardBoard FAQ

Hardware

- Processor : TMS320DM355 Processor
- NAND Flash : 2Gb Micron NAND Flash (256MB)
- DDR2 SDRAM : 1Gb Micron DDR2 SDRAM (128MB)
- Camera Interface :VGA CMOS Image Sensor by Default, Optional: 1.3M, 2M, 3M and 5 Megapixel CMOS Sensors supported
- Audio In/Out :AIC3104I Audio chip, Stereo Audio In/Out, 2.5mm Stereo Plug connector
- JTAG Port : JTAG Debugging Port
- Serial Port : RS-232 UART Debugging Port, 2.5mm Stereo Plug connector
- USB 2.0 Port : USB 2.0 HS (Device can be powered by USB port)
- SD Card Slot : SD/MMC Slot
- Power Management: TPS65053 Power management chip
- Power Input : +5V Power Input, 2.1mm I.D. x 5.5mm O.D. x 12mm Female
- Video Output Port : PAL/NTSC Output
- Network Interface : 10/100 Ethernet
- LCD/DVI Interface : LCD/DVI Interface through adapter board
- Expansion Port (NotPopulated) : SD/MMC, I2C, UART, McBSP, GPIO, 3.3V power supply
- PCB Board Mechanical : 3” x 2.5” (76.2mm x 63.5mm)



Distributions

- RidgeRun Free SDK customized for LeopardBoard. The are RidgeRun LeopardBoard SDK Hints available if you encounter a problem. A supported SDK is also available from RidgeRun that include hardware accelerated GStreamer audio / video support.

Production modules

- Lionboard 368 Low cost production ready SODIMM module 100% compatible with Leopardboard. Reduces TTM of video applications reusing preexisting distributions.

Category:

- ARM Development Boards

> From: eLinux.org

Micro2440

Micro2440 overviewFrom Rss feeds Soar so that you can: routing, lookup Subject matter [hide] A singleStamps Module 3 Specs: Seal of approval Unit 3 or more Specifications: SDK-Board Four Choosing the perfectSeal of approval Unit

FriendlyARM Micro 2440 Stamps Portion together with 4 hundred MHz New samsung S3C2440 ARM9 cpu.

Micro2440 Schematics

Standards: Stamp Segment

Dimensions: 63 back button Fifty two millimeter Pc: 600 MHz Samsung S3C2440A ARM920T (utmost freq. 533 MHz) RAM: Sixty-four MB SDRAM, 32 little bit Coach bus Thumb: Sixty-four MB 128 Megabytes 256 Megabytes Versus 1GB NAND Adobe flash and a pair of MB Neither of them Pen along with BIOS Successive, SPI, Universal serial bus, Liquid crystal, CMOS Digital camera Program Analog Knowledge as well as End result Consumer Outputs: 4 times Led lights Enlargement headers (2.Zero millimeter) Debug: Ten green JTAG (3.3 millimeter) OS Assist Home windows CE 5 various as well as Six A linux systemunix 3.Some Google android Specification: SDK-Board

http://www.china-pbx.cn/friendlyarm/Micro2440/

http://www.voice-logger.com/arm/Micro2440/

> From: eLinux.org

Mini210

mini210 : Equip Cortex-A8 S5PV210 1Ghz together with 5 " TFT Touchscreen display

Product or service Information and facts

Mini210 is really a Cortex-A8 high-performance advancement plank. The idea makes use of Samsung S5PV210 for the reason that key brand, jogging in up to 1GHz. PowerVR SGX540 S5PV210 a high-performance incorporated graphics website, help regarding 3D images operate correctly, and large smooth 1080P movie play back.

Mini210 utilize TFT Live view screen 5 " Resistive Touch-screen , 512M DDR2 memory space , 1GB SLC NAND Thumb, SD WiFi segment internal , Group Deb amplifier's WM8960 sound recording chip (can easily primary end result to helplecturer) . Mini210 likewise have miniHDMI HD end result, Browse Two.Zero, front and rear dslr camera, 8x8 matrix keyboard set user interface, assistance low-power life power signal power-saving function .

It is very suitable for establishing high-end MID, Android operating system Pc tablet , etc . Together with the Superboot , Person can easily super easy plus convenient to use just the micro sd card pertaining to replace or maybe use a various methods.

http://www.china-pbx.cn/friendlyarm/Mini210/

http://www.voice-logger.com/arm/Mini210/

> From: eLinux.org

MINI2440v2 developmentboard

(This page should probably be merged with Mini2440 ?)

(Admin: Does the author of this page know anything about wikipedia markup? It doesn't appear so. I've fixed up what I can, but all these pages by Whhenyuan need a lot of cleanups)

Contents

Development board

S3C2440

1.ARM9

- MINI2440v2
- SKY2440v2

S3C6410

2.ARM11

MINI6410

The S3C2440 includes the following components:

- separate 16 KB instruction and 16 KB data cache
- MMU to handle virtual memory management
- TFT& STN LCD controller
- NAND flash boot loader, system manager (chip select logic and SDRAM controller)
- 3-ch UART
- 4-ch DMA
- 4-ch timers with PWM
- I/O ports
- RTC
- 8-ch 10-bit ADC and touch screen interface
- camera interface
- AC97 audiocodec interface
- IIC-BUS interface
- IIS-BUS interface
- USB host, USB device
- SD host & multimedia card interface
- 2-ch SPI and PLL for clock generation.

Development Boards with MINI2440v2 with 3.5

Development Boards with MINI2440V2

[1]

Development Boards with MINI2440V2

Product Name	Company Name	Year Introduced	RAM MB	Flash MB	Processor	Source	Download Area
[2]	2009	up to 256	up to 512	i.MX27 400MHz + Spartan3a	[3]	Developer Site	

Beside the powerful CPU module, The MINI2440 development board also features:

- SDRAM
  - 64M SDRAM on board
  - 32 bit data bus
  - SDRAM clock frequency can reach upto 100MHz
- Flash Memory
  - 64M Nand Flash on board, nonvolatile
  - 2M Nor Flash on board, nonvolatile
- LCD control
- Interfaces and Resources
  - One 10MM Ethernet RJ-45 interface (DM9000 ethernet chip adopted)
  - 3 serial ports
  - One USB Host
  - One USB Slave (B-type interface)
  - One SD card interface
  - One stereo audio output interface, one MIC interface
  - One 10pin (2.0mm space) JTAG interface
  - 4 user LEDs
  - 6 user buttons
  - One PWM control buzzer
  - One adjustable resistance, used for AD conversion test
  - One I2C bus AT24C08 chip, used for I2C bus test
  - One 20pin (2.0mm space) camera interface
  - RTC battery on board
  - Power supply interface, with switch and indicator
- System clock source
  - 12M passive crystal
- RTC
  - Internal real time clock, battery backed
- Expansion interfaces
  - One 34pin 2.0mm GPIO interface
  - One 40pin 2.0mm system bus interface
- Dimension
  - 100 x 100 (mm)
- OS supported
  - Linux 2.6.13
  - WindowsCE.NET 5.0

Categories:

- Development Boards
- NeedsEditing

> From: eLinux.org

NaviEngine

NEC's NaviEngine 1 is a SoC product based on ARM's ARM11MPCore. It's for car navigation systems that adopts the SMP (Symmetrical Multicore Processor). The chip delivers high-speed parallel processing performance of up to 1920MIPS at 400 MHz. In addition, NaviEngine1 has 2D and 3D graphics using the POWERVR SGX 535 graphics core by Imagination Technologies. Volume production is scheduled to begin in March 2009.

Contents

Hardware

Chip

A block diagram shows the internals of NaviEngine1 chip.

Eval board

An evaluation board is available:

- NEC uPD35001 System-on-chip (ARM11 MPCore x4)
- DDR2 SDRAM (256MB)
- NOR Flash memory (64MB)
- NAND Flash memory (256MB)
- Three on-chip UARTs
- LAN9118 Ethernet adapter
- On-chip LCD controller
- USBH2.0, ATA6, PCI, CSI, SPDIF, I2S, etc

It's available from Personal Media Corporation for 312.900 Yen. See specification page and (japanese) product sheet.

Software

Operating systems

The following operating systems are running on NaviEngine:

- eT-Kernel Multi-Core Edition
- Symbian (on multicore)
- WinCE

- Linux (?)

Demo running eT-Kernel on one CPU and WinCE on the three other ARM11 cores. Demonstrates graphic performance, too.

QEMU

QEMU emulation for NaviEngine is part of 2008.05 OpenSolaris. It emulates

- DDR2 SDRAM (256MB)
- NOR Flash memory (64MB)
- Three on-chip UARTs

See OpenSolaris release note section 1.2 Architecture for details and OpenSolaris installation section 6.1 Setup QEMU.

Resources

- NaviEngine 1, System LSI for SMP-Based Car Navigation Systems
- NaviEngine Multicore Platform and its role in the evolution of car navigation systems
- K2L MOST starter kit for NaviEngine

Category:

- Development Boards

> From: eLinux.org

Opensourcemid

Contents

Introduction

OpenSourceMID.org is founded by a group of engineers enthusiastic in modern technologies and open source software development and sponsored by Timll Technic Inc. in China. It is an open community to provide support and discussions for K7 MID or any other open source MID projects through this website.

OpenSourceMID.org, itself, is an Open Source project. This means, first, that we offer not only a product but a process, and second, that we depend upon the contributions of developers and endusers to make that process happen. The easiest way for you to help us out is to join the overall OpenSourceMID.org project by join OpenSourceMID Community.

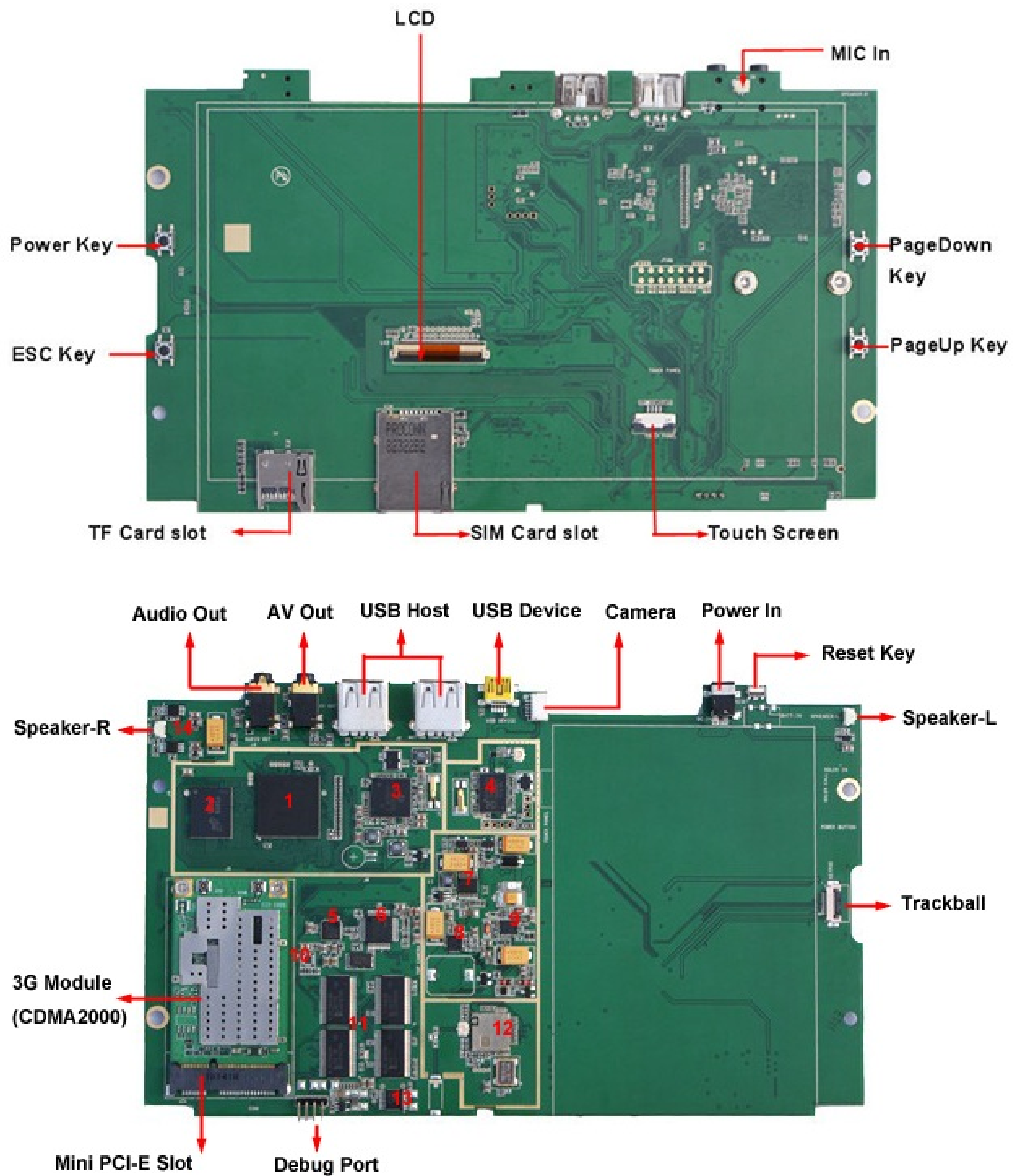
K7



K7 MID is a low-cost, high-performance handheld Mobile Internet Device featuring Texas Instruments' OMAP3530 ARM Cortex-A8 core processor with, wireless connectivity, touch screen control, GPS navigation systems and integrated power management with battery backup. It can support for both Android and WinCE operating systems and would be a complete solution that addresses all of the user's multimedia and communication needs on a single platform.

K7 schematic: Click here to download K7 MID operation manual: Click here to download (English) Click here to download (Chinese)

K7 mainboard



Number	Chip	Description	Datasheet
1	OMAP3530	The Processor	File:OMAP3530-datasheet.pdf
2	MT29C2G48MAKLCJA-6IT	256MB NAND Flash + 256MB SDRAM	File:MT29CxGxxMAxxxJA-6IT.pdf
3	TPS65930	Power Management	File:TPS65930-datasheet.pdf
4	GSC3f/LPx	GPS Chip	File:GSC3f-datasheet.pdf
5	USB3320	High Speed USB Transceiver	File:USB3320-datasheet.pdf
6	FE1.1	USB 2.0 High Speed 4-Port Hub Controller	File:FE11-datasheet.pdf
7	APW7093	Step Down DC/DC Regulator	File:APW7093-datasheet.pdf
8	TPS61032	Synchronous Boost Converter	File:TPS61032-datasheet.pdf
9	BQ24070	Single-Chip Charge and System Power-Path Management IC	File:BQ24070-datasheet.pdf
10	MMA8450Q	3-Axis, 8-bit/12-bit Digital Accelerometer	File:MMA8450Q.pdf
11	74ALVC164245	16-bit Dual Supply Translating Transciever	File:74ALVC164245-datasheet.pdf
12	WG7310	WLAN+BT+FM Module	File:WG7310-datasheet.pdf
13	TSC2046	Touch Screen Controller	File:TSC2046-datasheet.pdf
14	LM4890	1 Watt Audio Power Amplifier	File:LM4890-datasheet.pdf

Optional module - E380



E380 is a Mini PCI-E 3G wireless communication card based on EV-DO Rev.A standard，it is applicable to CDMA 2000 1x and EVDO network，it supports CDMA 800MHz and 1900MHz frequency band，it can achieve global roaming.

Product features: Size: 51mm(L)*30mm(W)*2.3mm(T) Working voltage: 3.3 - 3.8V DC Dual-antenna interface, support diversity antenna Support CDMA 2000 1x ，EV-DO Rev.0，EV-DO Rev.A Support CDMA 800/S1900 MHz

Performance parameters: Maximum output power: (Class III) BAND CLASS 0 (800/1900MHz)：+23dBm Minimum controlled output power: \<-50dBm Sensitivity: \<-104dBm (FER≤0.5％) Spurious maximum output power: 900kHz: \<-42dBc/30kHz 1.98MHz: \<-54dBc/30kHz Support high-speed data services CDMA 1x UL：153.6kbit/s(Peak) DL：153.6kbit/s(Peak) EVDO Rev.0 UL：153.6kbit/s(Peak) DL：2.4Mbit/s(Peak) EVDO Rev.A UL：1.8Mbit/s(Peak) DL：3.1Mbit/s(Peak)

Main functions: Support the China Telecom standard AT command set Support the connection speed and traffic statistics Support OTA function Fast-connect Support 8K EVRC Voice Code Embedded 3G wireless communication protocol stack

Optional module - E300



E300 is a CDMA2000 Internet card using USB port.

Product features: Size: 81*25.4*12mm Weight: 25.5g Dual-antenna interface, support diversity antenna Support CDMA 2000 1x ，EV-DO Rev.0，EV-DO Rev.A Support CDMA 800/1900 MHz

Performance parameters: CDMA 1x UL：153.6kbit/s(Peak) DL：153.6kbit/s(Peak) EVDO Rev.0 UL：153.6kbit/s(Peak) DL：2.4Mbit/s(Peak) EVDO Rev.A UL：1.8Mbit/s(Peak) DL：3.1Mbit/s(Peak)

Main functions: Support Micro-SD card(up to 16GB) USB 2.0 high-speed transmission

Environmental parameter: Working temperature: -20~55℃ Storage temperature: -30~80℃

Optional module - W338



W338 is a HSDPA Internet card using USB port.

Product features: Size: 87.9mm(L)*27mm(W)*12.05mm(T) Weight: 23g Chipset: Qualcomm MSM6280

Performance parameters: 3G frequency band: UMTS 2100MHz 2G frequency band: GSM/GPRS/EDGE 850/900/1800/1900 MHz

Main functions: Support Micro-SD card(up to 8GB)、USIM card management USB 2.0 high-speed transmission Single color LED can display a variety of state: real-time network state, working mode, error warning.

Environmental parameter: Working temperature: -20~60℃ Storage temperature: -30~85℃ Humidity≤93% Shake: 15g(Peak), 10~500Hz

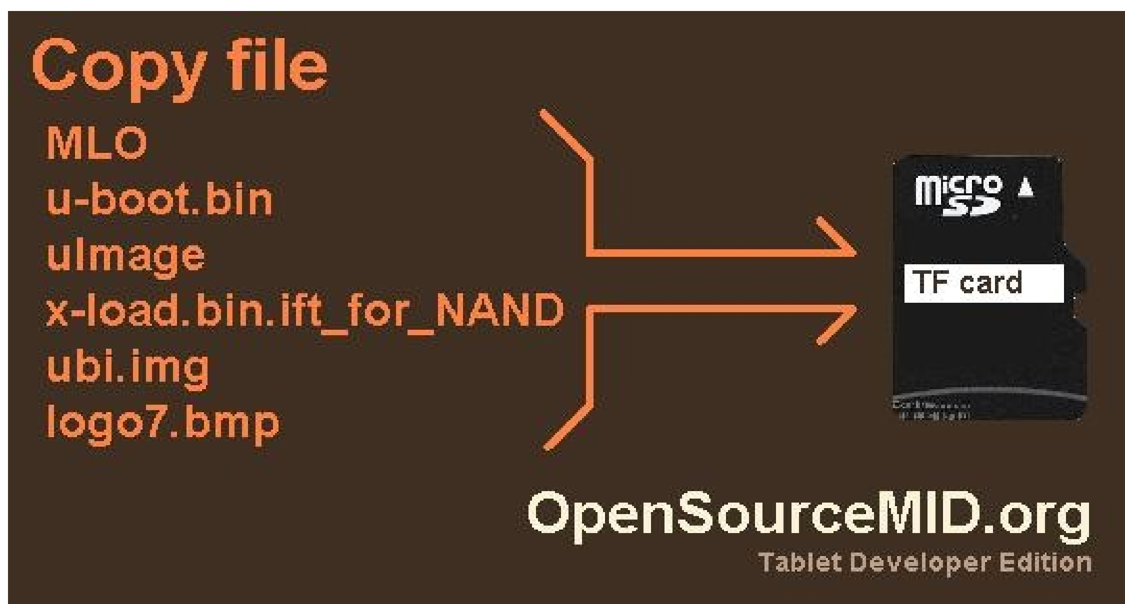
Android

How to update the Android

1.Please backup your TF data.

2 Use HP USB Disk Storage Format Tool(Click here to download) to format TF card.

3.Download the system image files from your backup data or website OpensourceMID.



4.You should insert the TF card before K7 is turned on.



5.Connect the 5V power supply, press the **Power key** and **Down key**, the LCD will display the interface below:

```
/============================================\
|                                            |
|                                            |
|      > update system                       |
|        update logo(logo7.bmp)              |
|        exit                                |
|                                            |
|--------------------------------------------|
|    up-key menu, down-key home,enter-key back |
\============================================/
```

Press the enter key to select the "update system", the system updating will start.

6.Select the "exit", the system will reboot.

How to compile the Android source

Please download the pdf from K7_Android_Development_Manual, you can find the details on the pdf. Chinese version pdf：Click here to download.

Opensourcemid Googlecode

How to use the USB device port as the UART debug port

1. Download the USB driver for Windows "Littleton USB Driver for Windows.rar" and extract it.

    i. Connect K7 to your computer's USB port. Windows will detect the device and launch the Found New Hardware wizard. Vista/7 users need to cancel the automatic driver update.
    ii. Select "Locate and install driver software."
    iii. Select "Browse my computer for driver software."
    iv. Click "Browse..." and locate the folder where you extracted the driver. As long as you specified the exact location of the installation package, you may leave "Include subfolders" checked or unchecked—it doesn't matter.
    v. Click "Next." Vista/7 may prompt you to confirm the privilege elevation required for driver installation. Confirm it. The driver will be installed.
    vi. Download the Android SDK for Windows "android-sdk_r07-windows.zip" and extract it.
    vii. Run the cmd in Windows.
    viii. Enter the commands below:

D: cd D:\android-sdk-windows\tools adb.exe shell

Now the USB device port is working as the UART debug port. You can install applications from computer. Enter the command below:

```
adb.exe install D:\AdobeReaderv10.0.0.apk
```

WinCE

How to update the WinCE

1.Please backup your TF data.

2.Use HP USB Disk Storage Format Tool(Click here to download) to format TF card.

3.Copy the following files in the backup data[\wince_6_R3\image] to the root directory of the TF card.

```
MLO
EBOOTNAND.nb0
NK.bin
XLDRNAND.nb0
```

Then change the EBOOTNAND.nb0 name to EBOOTSD.nb0 in the TF card.

4.Insert the TF card to K7 TF slot before power on.



1. long press Power Button and Esc Button, system will enter the update menu.

K7 MID

Update firmware? (detect that there are firmware in TF card)
1. Long press the Esc button to update.
2. Press the PageUp button to boot system normally.

6.Follow the instruction, and long press Esc Button, system will update automatically.

7.When finish updating, system will boot automatically.

From: eLinux.org

OSK

Contents

OMAP 5912 Starter Kit





OMAP5912 Starter Kit is a development board available from Spectrum Digital for $US295. It does not include a display but there is a Q-VGA LCD Module (below) and a Wire Wrap Prototyping Module

LCD Modules

Q-VGA from Mistral





Mistral Software Q-VGA LCD Module for OMAP5912 Starter Kit.

Note: a hardware modification is required to use the wakeup button.

Aditec Graphic LCD Module(AGLM)

Aditec Graphic LCD Module (AGLM)

From the website:

- CMOS Sensor Input, LCD Output
- Touch Screen, Navigation Switch
- Camera Input(2 Channel), TV Output

Files

The Angstrom distribution provides a replacement kernel and root file system for the OSK, based on the 2.6 kernel.

http://www.angstrom-distribution.org/releases/2007.12/images/omap5912osk/

Instructions for flashing these files:

http://ossie.wireless.vt.edu/trac/wiki/OSKNotes

Please report bugs to: http://bugs.openembedded.org/ and include omap5912osk in the bug report.

Contact philip at balister dot org with questions.

Links

- http://www.ti.com/omap5912osk
  - purchase from TI for $US295
  - Purchase from Spectrum Digital for $US295
  - Wire Wrap Prototyping Module
    - for $US99
  - USB gender-bender
    - get one of these to use the USB client mode, or a different nonstandard part: a USB A-to-A cable
  - Q-VGA Module from Mistral Software - for $US799
- http://omap.spectrumdigital.com/
- http://focus.ti.com/docs/general/splashdsp.jhtml?&path=templatedata/cm/splashdsp/data/linux_com_downloads
- MontaVista Linux Preview Kit - sign up and download (Linux 2.4)
- TI's OMAP Linux website has goodies including:
  - 2.6.8-rc3_080804.tar.bz2
    - a relatively old snapshot of the 2.6 OMAP kernel (warning: ~150MB download)
  - u-boot binaries and source - note that old versions use the Innovator-1510 board id, but U-Boot 1.1.2 should is used with 2.6
  - rootfsosk.tar.bz2
  - toolchain
  - omap5912osk_2.6.pdf
    - How to Build Linux for the OMAP5912 Processor
- For current code (2.6.13-rc1-omap1 at this edit) install GIT and run the following command:
  -
    - `git clone git://source.mvista.com/git/linux-omap-2.6.git`
      - (This might take a while, the tree is big)
    - or just get OMAP-Linux patches less current than the linux-omap GIT tree, against various older http://kernel.org kernels
    - you might be able to use kernel.org kernels directly, but some important drivers haven't yet made it upstream
- Use buildroot to quickly make a toolchain and rootfs
- http://oskfordummies.hp.infoseek.co.jp/
  - helpful OSK info
- http://oskfordummies.hp.infoseek.co.jp/howto/audio.html
  - audio help
- `#OL` on irc.freenode.net is an omap-linux dedicated IRC channel, and has some OSK developers. See http://irchelp.org/ for information on IRC.

Hardware Features

- Texas Instruments TMS320C55xx DSP core operating at 192 Mhz.
- ARM926EJ-S core operating at 192 Mhz.
- TLV320AIC23 codec with mic-in, line-in, and headphone jacks
- 32 Mbyte mobile DDR RAM

- 32 Mbyte on board NOR Flash ROM
- 4 Expansion connectors (bottom side)
  - Compatible with Mistral's Q-VGA LCD/touchscreen card
  - Compatible with Spectrum Digital's OSK wire Wrap Prototype Card
- RS-232 serial port
- 10 MBPS Ethernet port
- USB host port (usable as peripheral, with nonstandard cabling)
- CompactFlash socket (type I or II), for storage ONLY (no WLAN etc)
- On board IEEE 1149.1 JTAG connector for optional emulation
- +5 Volt operation, power supply included
- Size: 5.55" x 3.54" (141 x 90 mm), 0.062 thick, 8 layers

Software Features

- Compatible with Monta Vista Linux for OSK5912
- Compatible with OMAP Code Composer Studio from Texas Instruments
- Runs Linux 2.6 quite nicely
  - Current development is focused on the 2.6 kernels
  - Note that you should first install u-boot-1.1.2.

What's Included

- OMAP5912 OSK board
- Monta Vista Linux 2.4 for OMAP5912 OSK
- Documentation CD
- RS-232 cable
- Ethernet cable
- Audio cable
- AC Power cord(s) and Power supply
- Quick Start Guide

Flash Recovery Utility

- (free as in water) Flash Recovery Utility for the OSK5912 allows reflashing over USB - works with revision C boards
- (free as in speech, GPL) OMAP Flash Loader - link on bottom of Flash Recovery Utility
- ($79/$89 USD, supported product) FlashRecoveryUtility for OSK5912 from SDS Inc - works with revision C and D boards

HOWTOs and FAQs

This section tries to summarize various HOWTOs and FAQs existing for OMAP Linux and/or OSK.

- General OMAP Linux readme, also available in each kernel tree in Documentation/arm/OMAP/README
- OMAP GPIO usage HOWTO, also available in each kernel tree in Documentation/arm/OMAP/gpio
- Collection of OSK HOWTOs
- UBoot for OSK
- Spectrum digital OSK5912 FAQ
- OSK5912 Audio driver
- USB Camera HOWTO
- X on OMAP tutorial
- OSK for dummies

- Building Linux for the Innovator Development Kit for OMAP Platform
- Building Linux for the OMAP5912 OSK
- OSK5912 Newbie Guide
- Readme GIT for OMAP
- Submitting OMAP patches
- Linux OMAP Open Source Mailing List, the searchable archive is available at http://marc.info/?l=linux-omap.
- LCD output from decompressor: If you care about the first impression your kernel makes :-) , or your device has no serial ports, you might want to have the decompressor output its messages through LCD, like on a VGA.
- Flash u-boot using USB OTG on H3 (sorry, not OSK, but H3 related ;))
- Suspend To Disk For ARM
- Execute kernel in place (XIP) from flash Kernel XIP

TODO list

- Flash Recovery Utility (FRU) available above is uses an undocumented usb interface and is only for windows
  - Zach Welch wants the protocol specs so a scriptable version can be created for Linux
  - GPL'd OMAP Flash Loader listed above duplicates the protocol, but does not document it.
- u-boot sources with correct arch number (515 as listed on http://www.arm.linux.org.uk/developer/machines/) should be available from http://u-boot.sourceforge.net; or use u-boot 1.1.1 with Media:u-boot-1.1.1-osk.diff
- OE should build complete flash image
- uClibc rootfs should be available
- find/post patch info for 2.4 (2.6 is listed above)
- Tim Riker thinks it would be nice to have a uclibc/tuxscreen style buildroot too. ;))

Categories:

- Development Boards
- OMAP

> From: eLinux.org

PandaBoard

The PandaBoard is an OMAP4430 (Cortex-A9) based low cost development platform.

Contents

Hardware

- OMAP4 (Cortex-A9) CPU based open development platform.
  - OMAP4430 Application processor
  - 1GB low-power DDR2
  - Display HDMI v1.3 Connector (Type A) to drive HD displays, DVI-D Connector (can drive a 2nd display, simultaneous display; requires HDMI to DVI-D adapter), LCD expansion header
  - 3.5" audio in/out and HDMI Audio out
  - Full size SD/MMC card
  - Built in 802.11 & Bluetooth v2.1+EDR
  - Onboard 10/100 Ethernet
  - Expansion: 1xUSB OTG, 2xUSB HS host ports, General purpose expansion header (I2C, GPMC, USB, MMC, DSS, ETM)
  - JTAG, UART/RS-232, 2 status LEDs, 1GPIO button

More details can be found here

- PandaBoard EA1 Front



A hi resolution picture of the PandaBoard EA1 front is available here: http://elinux.org/images/d/d4/Panda_front.jpg

- PandaBoard EA1 Back

Availability

PandaBoard are in production and can be ordered from Worldwide distributors. You can also easily identify the board you have using Board revision id

Rev A3

Latest version of the board.

Rev A1/A2

Details

Rev EA1/EA2

These were limited number of 'Early Adopter' boards that built prior to production versions. more details

Rev ES

There is now a PandaBoard-ES which includes an OMAP 4460 at up to 1.2GHz. Several important differences make it important (at the present time) that the MLO/u-boot/kernel be specifically crafted for the 4460. The thermal management is not in the mainline 4430 code as yet and could cause unwanted thermal problems. So BEWARE of running any code built for the non -ES PandaBoard on the -ES model.

Accessories

- Bamboo Board
- Trainer Board
- Acrylic case
- 10" LCD LVDS plug-and-play bundle with capacitance touchscreen and ambient light sensor

-  BeadaFrame BeadaFrame 7" LCD display kit
  - 7" 800x480 TFT LCD screen
  - PWM Backlight control
  - Resistive touch panel
  - RTC time keeper
  - Plastic frame
-  Inti Media for Pandaboard Inti Media an expansion board with 10" LCD - watch in action here (multipoint capacitive touch screen) or here (resistive touch screen)
  - 10.1 WXGA LCD with multipoint project capacitive touch panel
  - 10.1 WSVGA LCD with resistive touch panel
  - 4 additional USB ports
  - 5 user controlled LEDs
  - 5 user controlled push buttons
  - expansion ports with 3V3 I2C, SPI, Serial

Recommended Reading Material

- Embedded_Linux_System_Design_and_Development
- Embedded_linux_primer
- Building_Embedded_Linux_Systems
- Essential_Linux_Device_Drivers
- Linux_Device_Drivers

How To's

- Build SDR for Pandaboard
- Add a GPIO Button
- How to build MLO and u-boot for the PandaBoard **-->> Updated 7/23/2011**
- How to build Barebox for the PandaBoard **-->> Updated 11/12/2011**
- How to build u-boot and SPL for PandaBoard and PandaBoard ES **-->> Updated 12/27/2011**
- How to build 3.2 kernel for the PandaBoard **-->> Updated 2/18/2012**
- How to build 3.3-rcx kernel for the PandaBoard **-->> Updated 2/18/2012**
- How to build 3.3 kernel for the PandaBoard **-->> Updated 3/19/2012**
- How to build 3.4 kernel for the PandaBoard **-->> NEW 6/3/2012**
- How to build 3.5-rcx kernel for the PandaBoard **-->> Updated 7/16/2012**
- How to add two additional USB connections to the PandaBoard

- How to add a power switch to the PandaBoard **-->> WIP**
- PandaBoard Power Measurements
- Miscellaneous Tips for using the Pandaboard under Ubuntu
- Build the latest kernel and run it on Pandaboard

Older How To's

- How to build a minimal file system using buildroot for the PandaBoard
- How to build a kernel using the buildroot toolchain from above for the PandaBoard
- How to build 2.6.38-rcx kernels for the PandaBoard **-->> Updated 3/15/2011**
- How to build 2.6.38 kernel for the PandaBoard **-->> Updated 3/16/2011**
- How to build 2.6.39-rcx kernel for the PandaBoard **-->> Updated 5/12/2011**
- How to build 2.6.39 kernel for the PandaBoard **-->> Updated 5/27/2011**
- How to build 3.0 kernel for the PandaBoard **-->> Updated 7/29/2011**
- How to build 3.0-rcx kernel for the PandaBoard **-->> Updated 7/11/2011**
- How to build 3.1-rcx kernel for the PandaBoard **-->> Updated 10/5/2011**
- How to build 3.1 kernel for the PandaBoard **-->> Updated 10/31/2011**
- How to build 3.2-rcx kernel for the PandaBoard **-->> Updated 2/18/2012**

Community

- Website: http://pandaboard.org
- IRC: #pandaboard @irc.freenode.net
- Mailing List: pandaboard@googlegroups.com
- panda on linux-omap@vger.kernel.org

Categories:

- OMAP
- Omap4430
- PandaBoard
- Development Boards

> From: eLinux.org

R-Pi Hub

(Redirected from RaspberryPi) Redirect page

Redirect to:

RPi Hub

> From: eLinux.org

RPi Hub

(Redirected from RaspberryPiBoard)

***Notice:** The Raspberry Pi Wiki pages on this site is collaborative work - the Raspberry Pi Foundation is **not** responsible for content on these pages.*

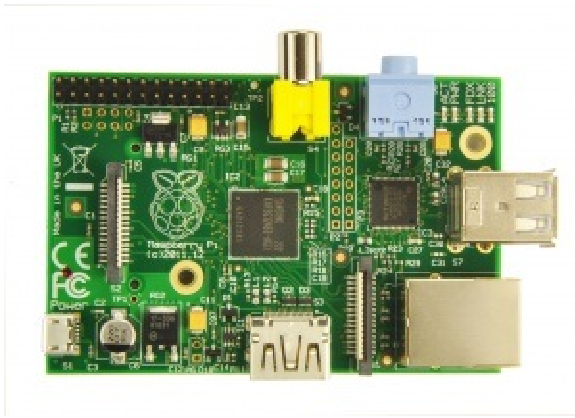
Contents

Raspberry Pi Wiki Hub deu | eng | fra | pt-br

Now shipping to customers

See the Buying Guide on how to order one, or visit the Raspberry Pi Foundation Home Page

About



The Raspberry Pi production board (model B Rev 2.0)

The Raspberry Pi (short: RPi or RasPi) is an ultra-low-cost ($20-$35) credit-card sized Linux computer which was conceived with the primary goal of teaching computer programming to children. It was developed by the Raspberry Pi Foundation, which is a UK registered charity (Registration Number 1129409). The foundation exists to promote the study of computer science and related topics, especially at school level, and to put the fun back into learning computing. The device is expected to have many other applications both in the developed and the developing world (Read more).

Raspberry Pi is manufactured and sold in partnership with the worldwide industrial distributors Premier Farnell/Element 14 and RS Components, and the Chinese distributor Egoman Technology Corp[1].

- You can get the latest news from the Foundation Home Page, the Twitter Feed or in the forums.
- For Raspberry Pi frequently asked questions see the FAQ section or the Raspberry Pi Foundation's FAQ page.
- Both manufacturing partners provide community areas for more technically focused discussions, articles, FAQs and related information:

- Premier Farnell: Element 14 Raspberry Pi Group

- RS-Components: DesignSpark - Raspberry Pi

- Products are RoHS, CE, FCC, CTick, CSA and WEEE compliant[2]. In common with all Electronic and Electrical products the Raspberry Pi should not be disposed of in household waste. Please contact the distributor from whom you purchased your Raspberry Pi device for details regarding WEEE in your country.

- Price: 20USD Model A+, 35USD for Model B+, excluding taxes, postage and packaging. For information about availability and shipping see the Buying Guide.

History

If you are interested in why the Raspberry Pi was created, and why it is what it is, check the General History page, which highlights relevant events in its history. It is not intended to be a detailed history, so it can be read quickly. You could also check the design changes page for how the Raspberry Pi has evolved, and the manufacturing differences page that may help if you are having problems with your board.

Getting Started

Buying Guide

Where can I get one and for how much?

- Base price is $20 for the model A+ and $35 for the Raspberry Pi 2. This price excludes local taxes and shipping.
- The Raspberry Pi's official worldwide distribution partners are Premier Farnell/Element 14 and RS Components
- Detailed information and other resellers can be found on the Buying Guide page.
- You can find out which peripherals and such are tested to work with the Pi in the Verified Peripherals section

Basic Setup

First little Raspberry Pi Steps...

- Ensure you have all the equipment you need to go with your Raspberry Pi.
- Become familiar with the board layout and connect it ready for power up.
- If you have not been provided with a pre-setup SD card you will need to prepare one with your chosen Operating System distribution
- If you are not using a HDMI monitor you may need to set up the correct video mode by editing the RPiconfig text file on the SD-card.
- Note: On the Debian OS after you log in you need to type **startx** at the prompt to get a graphic desktop.
- Particularly after first boot its important to do a clean shutdown with the command **sudo halt**
- **Having problems? Try the Troubleshooting page.**

Beginners Guide

You've just got your new Raspberry Pi device - what now?

- Beginners Guide
- Learn about the basics with the H2G2 - Introducing the Raspberry Pi entry.
- Read a small book for the Raspberry Pi Beginner [1]
- Get started with some basic projects and tutorials:
  - Raspberry Pi YouTube Tutorials
  - Raspberry Pi IV Beginners
  - My First Raspberry Pi Game
  - Guides, tutorials, tools and distribution downloads
- Easy GPIO Hardware & Software - in-progress at the moment
- Take a look through the Community section, which contains a range of beginner and advanced tutorials and guides, as well as groups to help you find like-minded developers.
- Pick up a copy of the Raspberry Pi Handbook to get you started on some fantastic projects
- Get started with Linux: Linux Basics

Resources

Hardware & Peripherals

- The Model B is more advanced than the Model A -

Software & OS Distributions

The Raspberry Pi will run a range of OS Distributions and run a variety of software.

- See Software for an overview, and OS Distributions for supported operating

Documentation

Datasheets

IC Datasheets and schematics links page.

see RPi Hardware.

- The RPi can be plugged into a suitable TV or monitor.
- The unit will support a range of devices, peripherals and accessories.
- The Low-level interfaces allow the use of optional Expansion Boards in a wide range of projects.
- The Foundation has launched a camera module with a 5MPixel sensor capable of capturing video at 30fps at 1080p
- Serial port connection instructions
- GPIO interface circuits for connecting switches, relays, etc
- For more advanced issues including see Advanced Setup.
- Setting up peripherals - examples/HowTos
- List of boards and user feedback
- Power Supply construction - HowTo
- Comparison to other hardware

system and pre-configured 'images'.

- Officially supported OS distributions include Raspbian, Arch Linux and RISC OS Open.
- Many unofficial distributions are available on the Distributions page.
- Advice is also available if you want to compile a kernel, boot from the network using U-Boot, or test the Pi's performance.
- The Raspberry Pi supports a wide range of programming languages, with many tutorials available.
- Information about installing. specific applications is available through the link.
- Extensive (boot) configuration info (config.txt) is available here.
- Information about various utilities that can be used with your Raspberry Pi can be found here.

Troubleshooting

Head over to the troubleshooting page for help fixing common proble

Bugs

Head over to the bugs page for a list of known bugs.

RPi Model B 3D CAD files

Theses are various 3D CAD Versions in both RAR and ZIP.

- CATIA V5 RAR http://sdrv.ms/JqdhMb
- CATIA V5 ZIP http://sdrv.ms/LjyLGD
- ProE RAR http://sdrv.ms/KCv1hZ
- ProE ZIP http://sdrv.ms/KCvhxq
- STEP RAR http://sdrv.ms/KCvv7T
- STEP ZIP http://sdrv.ms/JMhv18
- SketchUp http://scc.jezmckean.com/item/581
- SketchUp8 http://sketchup.google.com/3dwarehouse/details?mid=327d6b1d8bd6130d6fbd6b70c7f1d3e0
- Eagle 5 http://www.raspberrypi.org/phpBB3/viewtopic.php?f=41&t=4457
- STEP and various formats at tracepartsonline
- http://www.tronetix.com/joomla/index.php/projects
- http://www.hycshop.com/raspberry-pi-c-2_17/
- Maya, Autodesk FBX, Wavefront ZIP http://www.raspberryconnect.com/idonethat/item/166-raspberry-model-download}

RPi Model B+ 3D CAD files

- STEP ZIP http://www.inti-innovations.co.uk/products/inti-media/i media-raspberrypi.html (Docs and downloads section)

Education Material

- The Raspberry Pi Education Manual(PDF) http://downloads.raspberrypi.org/Raspberry_Pi_Education_Man
- Raspberry Pi GPIO Pin Worksheet for free (re)use (Multi Forma http://raspberry.tips/raspberrypi-tutorials/raspberry-pi-modell-b-p gpio-pins-worksheet/

Books

- Heitz, Ryan. Hello! Raspberry Pi. Manning Publications (2015). pp.
    1. ISBN 9781617292453

Education Material

- The Raspberry Pi Education Manual(PDF) http://downloads.raspberrypi.org/Raspberry_Pi_Education_Manual.pdf
- RAspberry Pi GPIO Pin Worksheet for free use (Multi Format) http://raspberry.tips/raspberrypi-tutorials/raspberry-pi-

- RAspberry Pi GPIO Pin Worksheet for free use (Multi Format) http://raspberry.tips/raspberrypi-tutorials/raspberry-pi-modell-b-plus-gpio-pins-worksheet/

Community

Projects, Guides & Tutorials

- An important source of information and guides is the Official Forum.
- Get started by following some of the many Tutorials.
- Common tasks and useful tips are available through the Guides page.
- Projects can be found, and added to, on the Projects page.
- Raspberry Pi Datasheets can be found on the DataSheets page.
- Knowledgeable users may want to review and help out with project wishlist items on the Tasks page.
- There are many tutorials, example projects and guides in The MagPi Magazine - which is available free online or to purchase in printed form.
- Some more great projects and setup guides in the Raspberry Pi Handbook

Schools, Universities, Clubs & Groups

- The Raspberry Pi Foundation's aims include encouraging education. Several groups including Computing At School aim to bring Computing Science back into schools.
- Go to the Education Page to add your project and find helpful links.
- Raspberry Jams are a great way to meet other Raspberry Pi users, share ideas and tips and learn more. To find a Raspberry Jam near you, see the Raspberry Jam page.

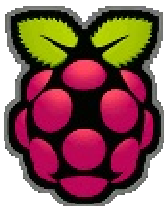
Supporting Communities

The Raspberry Pi Community is steadily growing:

- The Official Raspberry Pi Forum
- Element 14 Raspberry Pi Group, community site of Premier Farnell
- DesignSpark, community site of RS-Components
- 'Frambozenbier' (Raspberry Pi Homebrew)
- Stack Exchange Forum
- Non-official community of Raspberry Pi in spanish language
- World Of Pi A forum based on all things Raspberry Pi.
- The MagPi Magazine - Community based, free eMagazine, get involved!
- RaspberryPi Osdev - Hardware specific OS-development community, sitting in freenode.net#raspberrypi-osdev.
- news:comp.sys.raspberry-pi - Usenet newsgroup

About the RPi Wiki

Do not be afraid to add your bit, content is vital for the wiki to function.



A 3D rendering of the Raspberry Pi logo

Translations

The wiki is being translated into several languages, some of which can be seen on the hub banner above. Current languages include:

- English: R-Pi Hub

- Português: Pt-BR:R-Pi Hub
- 简体中文: Zh-CN:RPi_信息中心
- Deutsch: DE:R-Pi_Hub
- മലയാളം: Ml:R-Pi Hub

Any help translating would be greatly appreciated. Thank you to those who have already contributed!

References

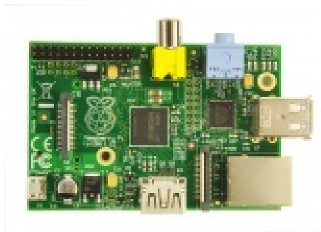
1. ↑ http://www.raspberrypi.org/archives/3195
2. ↑ http://www.element14.com/community/docs/DOC-44828/l/raspberry-pi-safety-data-sheet

3. v
4. t
5. e

Raspberry Pi

Startup

Buying Guide - SD Card Setup - Basic Setup - Advanced Setup - Beginners Guide - Troubleshooting



Hardware

Hardware - Hardware History - Low-level peripherals - Expansion Boards

Peripherals

Screens - Cases

- Other Peripherals (Keyboard, mouse, hub, wifi...)

Software

Software - Distributions - Kernel - Performance - Programming - VideoCore APIs - Utilities

Projects

Tutorials - Guides - Projects - Tasks - DataSheets - Education - Communities

Category:

- RaspberryPi

> From: eLinux.org

S3C2410

The S3C2410 is developed using an ARM920T core, 0.18um CMOS standard cells and a memory complier. Its low-power, simple, elegant and fully static design is particularly suitable for cost and power sensitive applications. Also, the S3C2410 adopts a new bus architecture, AMBA (Advanced Microcontroller Bus Architecture) An outstanding feature of the S3C2410 is its CPU core, a 16/32-bit ARM920T RISC processor designed by Advanced RISC Machines, Ltd. By providing a complete set of common system peripherals, the S3C2410 minimizes overall system costs and eliminates the need to configure additional components.

To reduce total system cost, the S3C2410 also provides the following features:

- separate 16KB Instruction and 16KB Data Cache
- MMU to handle virtual memory management
- LCD controller (STN & TFT)
- NAND Flash Boot loader
- System Manager (chip select logic, SDRAM controller)
- 3-ch UART with handshake
- 4-ch DMA
- 4-ch Timers with PWM
- I/O Ports
- RTC
- 8- ch 10-bit ADC and touch screen interface
- IIC-BUS interface
- IIS-BUS interface
- USB Host
- USB Device
- SD Host & Multimedia Card Interface
- 2-ch SPI
- PLL for clock generation.
- Clocked up to 203MHz

S3C2410 Users Manual

Product Page

- SBC2410-III Single Board Computer based on Samsung **S3C2410** from Embest
- Mini2410-II Processor Card based on Samsung **S3C2410** from Embest

Category:

- ARM processors

From: eLinux.org

SBC3530



EMA SBC3530——http://www.ema-tech.com

Contents

EMA SBC3530



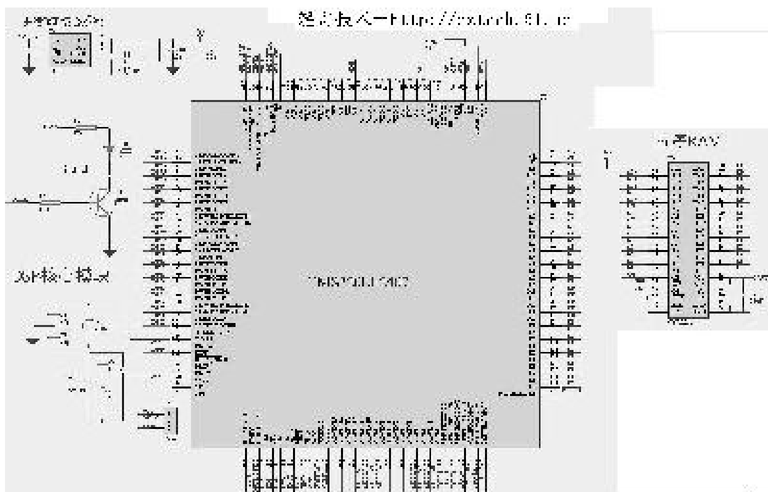
EMA SBC3530

Overview

1.SBC3530 from EMA is a Low-power, high-performance single board computer based on TI's OMAP35x multimedia applications processor. Designed with EPIC Standard and extendable architecture . SBC3530 helps users leverage the benefits of OMAP35x processor comprising of advanced Superscalar ARM Cortex-A8 RISC core with NEON co-processing, IVA2.2 subsystem with a C64x+ digital signal processor (DSP) core and SGX subsystem for 2D and 3D graphics acceleration (PowerVR). 2.The SBC3530 helps users building their products around the OMAP35x enjoy the benefits of faster user interfaces, faster data access and multi-standard encode/decode. Resolution up to 1920 * 1080, Supports multiple display output, such as VGA, LCD, S-Video and AV composite video output. Supports multi-standard encode/decode at D4 (1280×720p/60Hz) 30 fps. Supports multiple embedded operation system, such as Android, Embedded Linux, WinCE. Supports 4-channel video input, maximum support 4-way CVBS. Built-in hardware encryption

and hardware authentication mechanism to better protect the user's application software. 3.SBC3530 comes with X-Loader、U-Boot、Linux Kernel（V2.6.30） and JFFS2 File system for Linux. The PSP Software Release from TI includes the Code Sourcery tool chain and the required source code and utilities. 4.It is an ideal platform for users to develop their applications.Present SBC3530 has been applied to mobile Internet devices, Global Positioning System (GPS), 2D/3D Game Station, medical equipment, Image Capture Machine, HMI, etc.

Specifications



Hardware

CPU

TI OMAP3530 Multi-Core CPU

uperscalar ARM Cortex-A8 RISC core NEON™ SIMD co-processing 430MHz TMS320 C64x+ (DSP) core POWERVR SG™ 2D/3D graphics acceleration

RAM

128MByte/256MByte/512MByte DDR

Flash

128MByte/256MByte/512MByte/1GByte Nand Flash

Ethernet

1x 10M/100M High performance Ethernet, RJ45 interface

COM Port

1x RS232(5 Lines: TX,RX,CTS,RTS,GND)

2x 3.3V UART(5 Lines: TX,RX,CTS,RTS,GND)

USB Host

4x USB 2.0 high speed host(each provide 500mA current)

USB OTG

1x USB 2.0 high speed OTG（Device option）

Audio

1x 3.5mm stereo audio phone jack

1x 3.5mm microphone jac

VGA

Standard VGA, support all VESA standard resolution

AV/S-Video Video Output

Support mode list：

NTSC-J, M

PAL-B, D, G, H, I

PAL-M

CGMS-A

Video Input

4 channel video input

Support up to:

4 CVBS

Or 1 CVBS + 1 YPbPr/RGB(embedded sync)

Or 1 CVBS + 1 YPbPr/RGB(embedded sync)

Or 2 CVBS + 1 S-Video

Or 2 S-Video

CVBS supports NTSC/PAL/SECAM

SD Slot

MMC/SD/SDIO/SDHC slot, support up to 32GByte

FRAM

64KBit high speed F-RAM with virtually unlimited writing cycle

Security

Hardware security-dog for hardware & software protection

RTC

CR1220 RTC battery socket

Button

1x Side button (user interrupt)

1x Reset button (inside)

1x PowerON button (inside)

LED

2x Power LED

2x Power management unit LED

2x GPIO LED

Boot switch

1x 6Bit boot mode switch

Power

1x 5V 4A DC Jack

1x 5V 4A TJC3 connector

1x Battery module connector

1x Power management signal connector

Expansion connectors

1x 3.3V I2C port

1x 3.3V LCD interface with touch screen support

1x 1.8V SPI port

1x 1.8V SDIO expansion for WIFI module

1x 1.8V 6x6 scanning keypad connector

1x 1.8V GPIO expansion

1x 1.8V GPMC High speed bus expansion

1x 1.8V Digital camera interface

Operating conditions

Conditions	Min	Normal	Max
Power supply	5V 0.3A	5V 0.5A（mark 1）	5V 4A（mark 2）
power consumption	1.5W	2.5W（mark 1）	20W（mark 2）
Temperature(Commercial)	0°C	/	70°C
Temperature(Industrial)	-40°C	/	0°C

Mark 1：The value is the average detected in the condition that CPU running at full speed in 600MHz working condition, other circuit in a working conditon, and does not access other modules, USB interface don’t in working condition. Mark 2：The value is the average detected in the condition that 4 USB interface supply Power, and the board connected to the LCD module, digital camera, WIFI and other cases, etc.

Loading...



广州英码信息科技有限公司
Guangzhou Embedded Machine Technology Co.,Ltd.
Add: NO.152,XinGang West Road, Haizhu District, Guangzhou, China
Tel: 86-20-61230220 Fax: 86-20-61230221
Web: http://www.ema-tech.com

Categories:

- Linux
- OMAP
- Development Boards

> From: eLinux.org

SBC8100

OMAP3530多功能开发板-SBC8100

深圳市天漠科技有限公司在**11**月**12**日推出一批 **256MB DDR SDRAM 256MB Nand Flash**配置升级版 **DevKit8000** 评估套件，限量**100PCS**，欢迎来电咨询选购。

SBC8100单板机是深圳市天漠科技有限公司继DevKit8000后基于德州仪器(TI) OMAP35x系列处理器推出的又一款功能强大的多功能单板计算机。

SBC8100采用德州仪器（TI）OMAP3530处理器作为CPU。OMAP3530处理器集成了600MHz的 ARM Cortex™-A8 内核及430MHz的具有高级数字信号处理算法的DSP核，并提供了丰富的外设接口。SBC8100扩展出了 网口、 SD/MMC接口、串口、Audio IN/OUT、Camera、S-Video/AV OUT、VGA、WiFi、Bluetooth、GPS、TTL LCD、高速USB HOST、USB OTG、电源、keyboard、扩展接口、4个自定义Button及总线接口。为了适应多种场合下的应用，本产品采用了核心板Mini8100加扩展板SBC8100的分离式结构进行开发，其性能可靠，稳定性高，易于扩展。

此工程的建立主要是为用户介绍SBC8100的硬件软件资源及在开发OMAP35x时提供一个交流分享平台，可使用户快速的对SBC8100单板机进行全面的了解，并对开发的常见问题进行总结归纳。

Contents

特性概述





SBC8100单机版主板

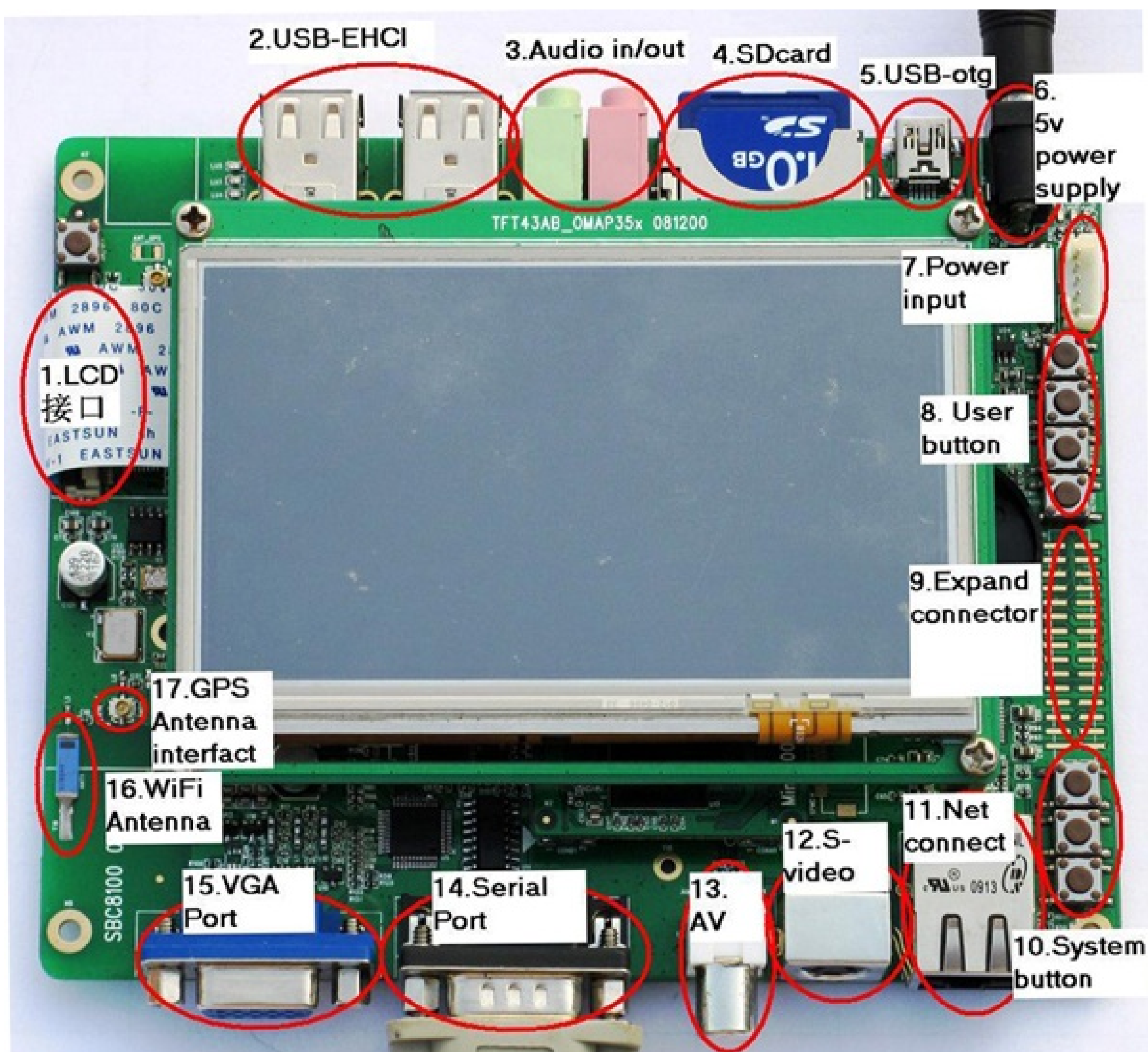
SBC8100 andriod+4.3"LCD屏

- SBC8100单板机采用德州仪器（TI）OMAP35x通用处理器作为CPU。OMAP35x处理器集成了600MHz的 ARM Cortex™-A8 内核及430MHz的具有高级数字信号处理算法的DSP核，并提供了丰富的外设接口。SBC8100单板机外扩了网口、**SD/MMC**接口、串口、音频输入输出接口、**Camera**、**S-Video/AV OUT**、**VGA**、**WiFi**、**Bluetooth**、**GPS**、**TTL LCD**、**USB HOST**、**USB OTG**、电源、**keyboard**、扩展接口、**4**个自定义**Button**及总线接口。
- SBC8100为开发者使用OMAP3530处理器提供了完善的软件开发平台，支持linux-2.6.22及WinCE 6.0操作系统，并包含完善的底层驱动程序，方便用户体验OMAP35x处理器强大的处理功能、设计系统驱动及其定制应用软件，并提供有成熟的操作系统Google Android及DVSDK Demo。完善的产品用户手册、驱动及芯片数据手册使用户更快的基于此主板对OMAP3530进行开发。

硬件特性

OMAP3530处理器集成了600MHz的 ARM Cortex™-A8 内核及412MHz的具有高级数字信号处理算法的DSP核，SBC8100单板机扩展了OMA3530的多种性能，具体的硬件接口特性如下所示。

接口图



硬件特性描述

- 处理器
  - OMAP3530 处理器（Pin to Pin兼容OMAP3503，OMAP3515,OMAP3525处理器）
  - 600-MHz ARM Cortex™-A8 Core
  - 430-MHz TMS320C64x+™ DSP Core
  - 集成存储器用于ARM CPU (16kB I-Cache, 16kB D-Cache, 256kB L2) 和片上存储 (64kB SRAM, 112kB ROM)
- 存储器
  - 128MByte DDR SDRAM
  - 128MByte NAND Flash
- 音频/视频接口
  - 一个S-VIDEO接口
  - 一个VGA输出接口
  - 一个TV OUT接口
  - 24bit真彩色LCD接口（含4线触摸屏接口,分辨率可支持2048x2048）
  - 一个音频输入接口
  - 一个2声道音频输出接口
- 传输接口
  - 3 x 5 线串行接口，RS232电平
  - USB接口：1 x USB2.0 OTG， High-speed， 480Mbps，4 x USB2.0 HOST， High-speed， 480Mbps
  - 2路SD/MMC接口，支持3.3V及1.8V逻辑电压
  - 网络接口：10/100Mbps，RJ45 connector

    - 2路McBSP接口（多功能串行接口）
    - 1路ULPI
    - 1路IIC
- 输入接口
    - 1个CAMERA接口（可外接CCD和CMOS的摄像头）
    - 4 X 5键盘接口
    - 1个启动引导按键
    - 1个Reset按键
- 电气特性
    - 核心板尺寸 : 59 mm x 37 mm
    - 主板尺寸：114.1mm x 114.9mm
    - 输入电压 : +5V
    - 功 耗 : 0.3.4A @ 5V
    - 工作温度 : -0 to 70℃(芯片支持)
    - 操作湿度 : 20% ~ 90%

芯片介绍

SBC8100单板机	芯片名称	备注
处理器芯片:	OMAP3530CUS	0.65mm的CUS封装
存储器芯片:	MT29C1G24MADLAJA-6IT	128MB mDDR/128MB NAND，Flash和SDRAM封装在同一个芯片
电源芯片:	TPS65930BZCH	外扩电源管理，RTC，USB OTG，音频，6X6键盘
DVI-D芯片	TFP410	S-VIDEO输出,输出DVI-D信号
网口芯片	DM9000	RJ45接口，10M/100M自适应，目前实测速度可达36M
串口芯片	MAX3232 CSE	RS232电平

接口介绍

扩展板

成品：

- 模拟摄像头输入模块CAM8000-A

标准720*576PAL制式分辨率；通过30PIN FFC排线连接Devkit8000，另一边通过BNC连接头连接摄像头设备，实现通过摄像头输入信息的功能。

- VGA高清视频输出模块VGA8000

基于PHILIPS 74alvc164245芯片，专为Devkit8000设计的可选配套液晶显示模块，240MHz的最大采样速度，可以输出标准的液晶显示屏信号，在分辨率高达1024*768下可以流畅显示

- USB WiFi无线模块WF8000-U

基于USB接口的WIFi无线通讯模块，适用于天漠所有带USB接口产品。该模块采用一种可以将个人电脑等终端以无线方式互相连接的技术，基于IEEE 802.11标准的无线网路通讯协议，高度集成 MAC / BBP和2.4GHz射频单芯片。它完全适应IEEE 802.11 b/g的高标准，无线连接范围大，有庞大的吞吐量。凭着优越的射频架构和优化算法，构建了WF8000-U WIFi模块良好的性能和低功耗消费

- GPS定位系统模块GPS8000-S

该模块采用最新表面贴片和先进的集成电路技术，以取得产品的最佳性能，同时也减小产品体积并将功耗减低到最小。综合硬件的高性能和软件的高智能度使模块具有更强的兼容性，并广泛应用于各种导航设备与导航产品中。

- GPRS通讯系统模块GPRS8000-S

基于GPRS的GSM/GPRS解决方案，使用工业标准界面，使其具备小尺寸、低功耗等诸多优点，可以实现语音、SMS、数据和传真信息的高速传输，可广泛用于WLL，M2M和各种手持设备。

显示方式

S-Video显示

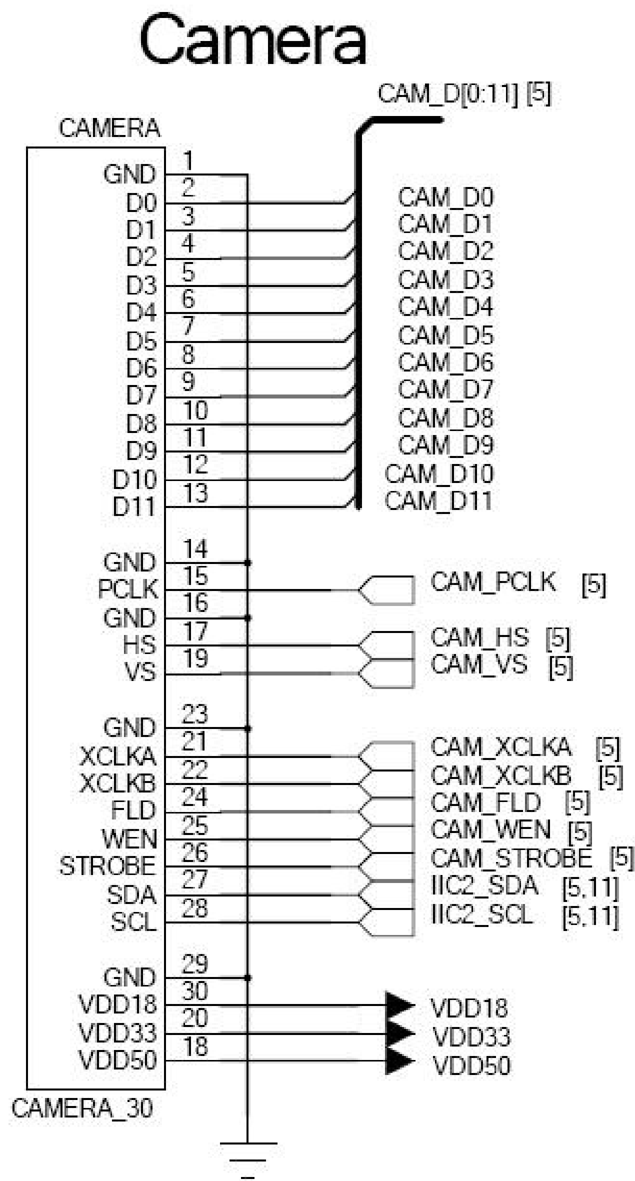
S-video四线接口，可输出视频信号，不包括音频信号，可外界S-Video接口的显示器。

VGA高清显示 可输出标准的液晶显示屏信号，在分辨率高达1024*768下可以流畅显示 **TV OUT**输出显示 可输入高清视频信号。

LCD屏接口 24bitLCD屏接口，可输出真彩色RGB信号，R:B:G=8:8:8,分辨率最大支持2048*2048. 接口为50-pin FPC 连接器，间距0.5mm。 接口信号类型：

RGB data信号	LCD控制信号	SPI信号	IIC信号	触摸屏信号	电压输出
24bit	6bit	4bit	2bit	4bit	5bit
R:G:B=8:8:8	行列等控制信号	标准spi信号	2位IIC信号	4线触摸屏	输出电压5V，3.3V，1.8V

摄像头接口



扩展接口

40PIn接口，间距2.0mm 特性如下所示：

引脚	信号定义	功能描述
1	GND	GND
2	BSP1_DX	Transmitted serial data 1
3	BSP1_DR	Received serial data 1
4	BSP1_CLKR	Received clock 1
5	BSP1_FSX	Transmit frame synchronization 1
6	BSP1_CLKX	Transmit clock 1
7	BSP1_CLKS	External clock input 1
8	BSP1_FSR	Receive frame synchronization 1
9	UART1_CTS	UART1 clear to send
10	UART1_RTS	UART1 request to send

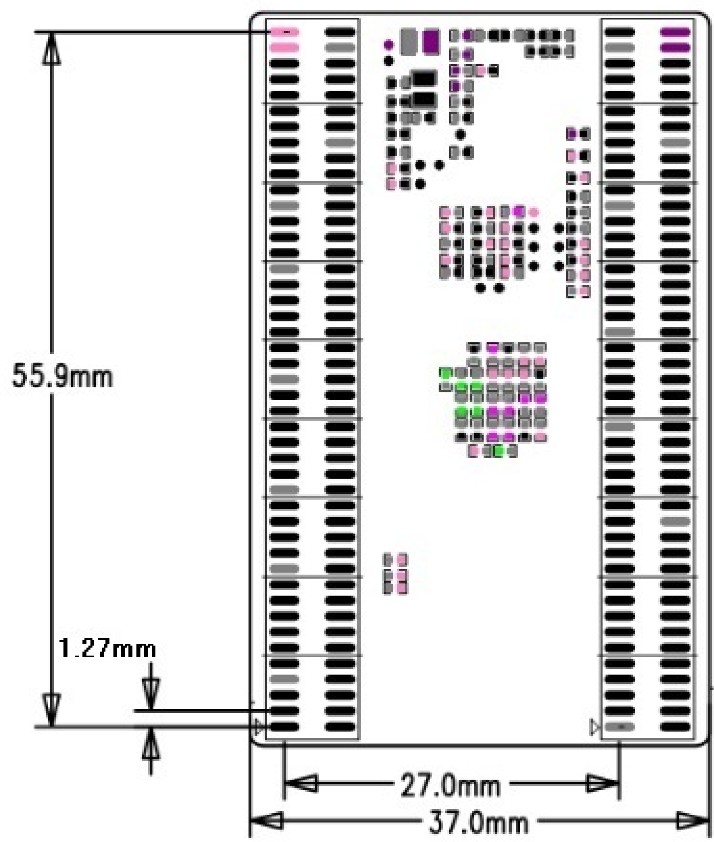
11	UART1_RX	UART1 receive data
12	UART1_TX	UART1 transmit data
13	GND	GND
14	MMC2_CLK	MMC2 card clock
15	MMC2_CMD	GND
16	MMC2_D0	MMC2 card data 0
17	MMC2_D1	MMC2 card data 1
18	MMC2_D2	MMC2 card data 2
19	MMC2_D3	MMC2 card data 3
20	MMC2_D4	MMC2 card data 4
21	MMC2_D5	MMC2 card data 5
22	MMC2_D6	MMC2 card data 6
23	MMC2_D7	MMC2 card data 7
24	BSP3_DX	Transmitted serial data 3
25	BSP3_DR	Received serial data 3
26	BSP3_CLKX	Transmit clock 3
27	BSP3_FSX	Transmit frame synchronization 3
28	GND	GND
29	IIC3_SCL	IIC3 master serial clock
30	IIC3_SDA	IIC3 serial bidirectional data
31	SPI1_SIMO	Slave data in, master data out
32	SPI1_SOMI	Slave data out, master data in
33	SPI1_CLK	SPI1 clock
34	SPI1_CS0	SPI enable 0
35	SPI1_CS3	SPI enable 3
36	HDQ_SIO	Bidirectional HDQ
37	VDD33	3.3V
38	VDD18	1.8V
39	VDD50	5V
40	VDD50	5V

LAYOUT

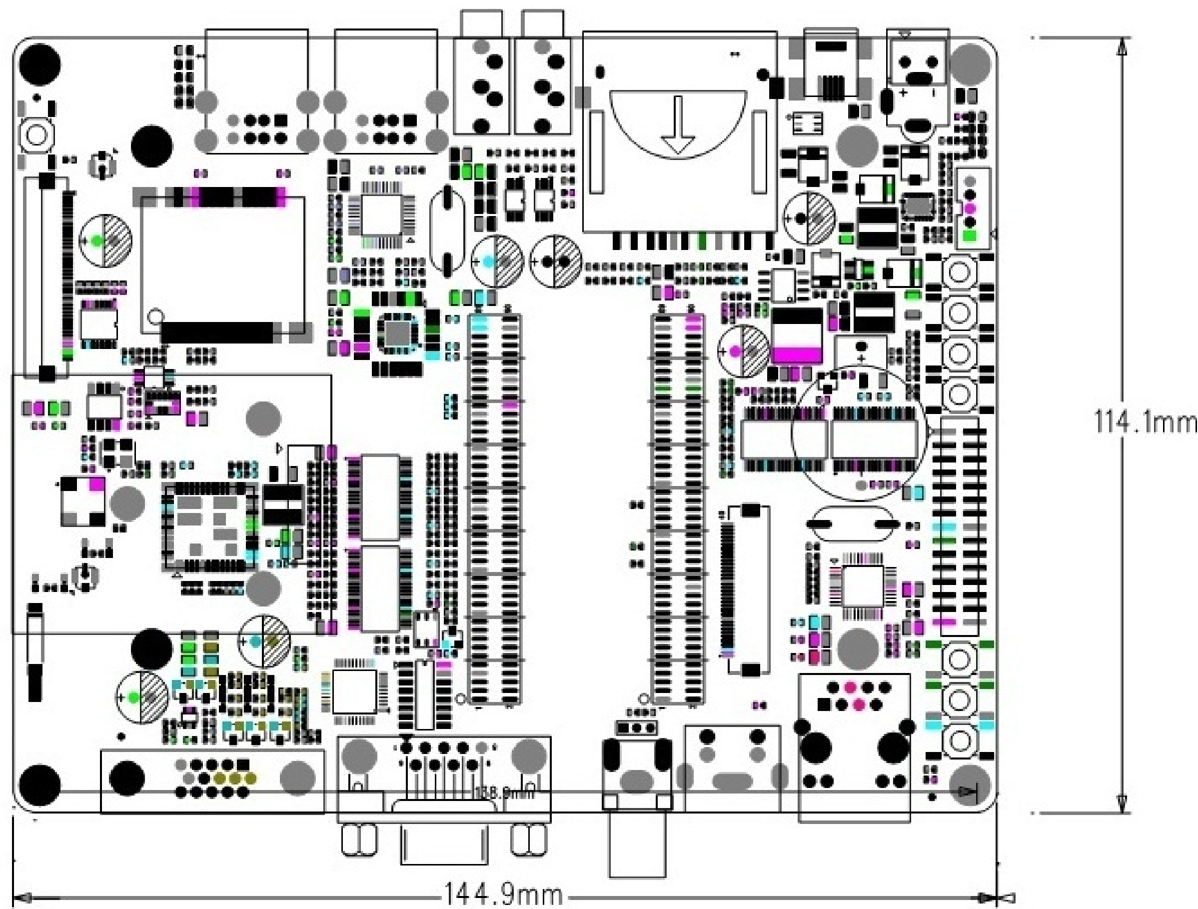
核心板采用六层设计，其中每层分布情况如下

1	2	3	4	5	6
表层	地层	信号	电源	地层	底层

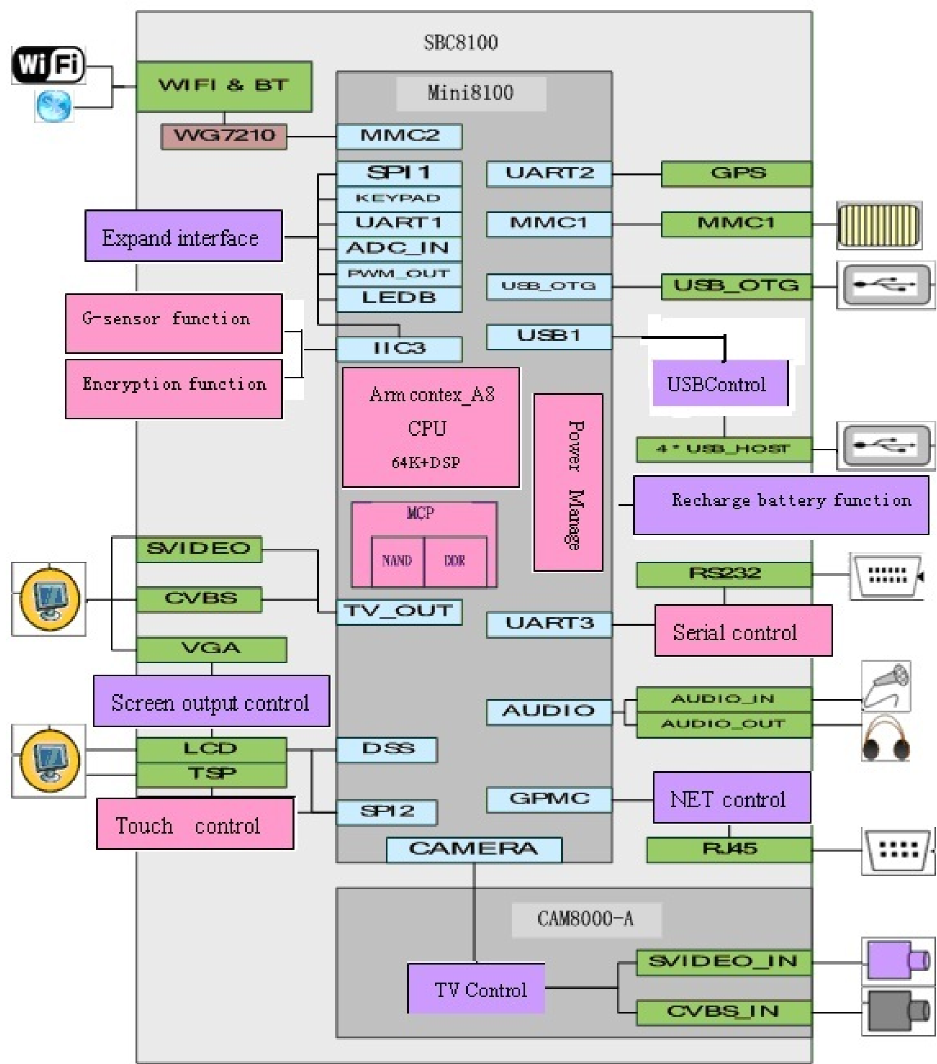
核心板尺寸图如下所示



.底板尺寸图如下所示



主板特性图



软件应用开发

软件特性

SBC8100多功能单板机支持WinCE6.0及linux2.6.22操作系统，具体软件特性请参考下表。

系统	项目	特性	描述
Linux	启动代码	版本	x-load-1.41，u-boot 1.3.3
		启动方式	支持网络、SD卡或NAND Flash中启动引导Linux系统
		映像更新	支持网络或SD卡更新映像
	内核与驱动	版本	Linux 2.6.22
		支持文件系统格式	ROM/CRAM/EXT2/EXT3/FAT/NFS/ JFFS2/UBIFS file systems
		驱动支持	Serial, RTC, Net, Flash, LCD, Touch screen, S-Video, TV out, VGA, Audio In/Out, SD, USB Host, USB OTG, Keypad, WiFi, GPS, LED
	文件系统	文件系统格式	Ramdisk文件系统、UBI文件系统
		系统特性	库支持（ALSA -lib, tslib, glibc）,udev设备管理支持
	Demo	Angstrom	音频（XMMS），网络（Firefox），图形编辑器（gimp）及文档处理软件(Abiword)
		Android	Google开发的基于Linux平台的开源手机操作系统
		DVSDK软件	支持MPEG4，MPEG2，H264，mp3，aac音\视频格式解码
WinCE	启动代码	版本	x-load-1.41、Eboot
		启动方式	支持网络、SD卡或NAND Flash中启动引导wince系统
		映像更新	支持网络或SD卡更新映像
	系统特性	内核特性	KITL内核调试, Reboot, Watchdog, RTC
		驱动支持	显示驱动（S-Video,AV, TFT LCD）
			Serial, RTC, Net, Flash, LCD, Touch screen, S-Video, TV out, VGA, Audio In/Out, SD, USB OTG, USB Host, Keypad, WiFi/BT, GPS, LED, VRFB, DSPLINKK/CMEMK, PWM, ADC, GPIO/I2C/SPI/MCBSP
		系统功能	电源管理（背光驱动、电池驱动、休眠\唤醒功能）
			HIVE注册表支持
			ROM文件系统支持
		软件特性	Media play 9.0, Word编辑工具及Internet Explorer 6.0
			.NET Compact Framework 3.5

linux开发

LINUX系统快速操作

1. 系统启动方法
2. 显示方式选择

LINUX系统开发

1. 开发环境搭建
2. 系统编译
3. 系统定制
4. 源码分析

Demo方案展示

Android

SBC8100已成功移植Android系统。

- SBC8100可运行基于Android系统的各种应用程序。
- SBC8100支持4.3"LCD，5.6"LCD及7"LCD图形界面交互及触摸屏功能.
- 可使用Android系统内置的音频播放器播放各种音频文件
- 可通过SD卡或USB OTG接口传输数据。
- 播放音频，浏览图片及一些基本的功能。

更详细的使用及移植DVSDK的方法，请参考SBC8100_Android



DVSDK（DSP）

SBC8100在linux下可基本运行TI提供的DVSDK包。
SBC8100支持的DVSDK包具有如下功能：

- 支持2D/3D图像加速功能
- 支持DSP编解码（可支持音频视频硬件解码）
- 支持s-video视频输出
- 可播放3D视频
- 硬件解码播放音频文件，格式支持：mp3，aac
- 硬件解码播放视频文件，格式支持MPEG4，MPEG2，H264

更详细的使用及移植DVSDK的方法，请参考SBC8100_DVSDK



Wince系统

套件概述

产品特性

SBC8100单板机是一个多功能开发平台，为嵌入式设计人员提供快捷简单的实践方式来体验OMAP3530处理器强大的处理和运算能力。该单板机提供了一个完整的开发平台，包括一个4.3"LCD屏（分辨率为480x272）、SD卡、电源及各种接口转接线（串口，S-Video）等。该产品使设计者能够基于SBC8100的板载系统快速开发出基于OMAP3530芯片的相关产品。

SBC8100多功能单板机为开发者使用OMAP3530处理器提供了完善的软件开发平台，支持linux-2.6.22及WinCE 6.0操作系统，并包含完善的底层驱动程序，方便用户快速体验OMAP35x处理器、设计系统驱动及其定制应用软件，并提供有成熟的操作系统Google Android及DVSDK Demo。完善的产品用户手册、电路原理图及芯片数据手册使用户更快的基于此主板对OMAP35x进行开发。 SBC8100多功能单板机的配件如下所示：



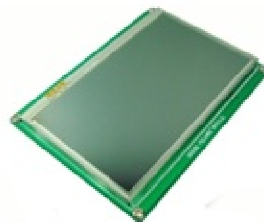






网线　USB转接线 A转mini-B　串口转接线　S-video转接线　SD卡　电源适配器






DevKit8000评估主板　CAM8000-A+ 摄像头　4.3"LCD（480*272）+触摸屏　产品光盘

SBC8100单板机分两种配置：标准配置和可选配置。

标准配置：包含完善的接口配件的支持，具备了、S-Viode线、USB线等相关配件，该配置主要针对特定应用的专业产品开发人员。 完全配置：包含模拟摄像头输入模块CAM8000-A

产品型号	配件清单
SBC8100标准配置 标准配置	•SBC8100单板机 *1 •SD卡（512MByte） *1 •交叉串口线 *1 •网线 *1 •5V 2A电源 *1 •Mini USB B线转USB A型公头 *1 •S-Video线 *1 •光盘(Linux源码驱动、WinCE BSP源码、底板原理图、WinCE应用原代码、接口驱动等) *1
SBC8100单板机可选配件	•7"LCD屏（LCD8000-70T，分辨率为800*480） •5.6"LCD屏（LCD8000-56T，分辨率为640*480） •4.3"LCD屏（LCD8000-43T，分辨率为480*272） •模拟摄像头模块 CAM8000-A •模拟摄像头模块 CAM8000-A

光盘特性

光盘中包含如下内容，

- 软件

linux驱动：Serial、RTC、Net、Flash、TV OUT、LCD、音频、触摸屏控制器、MMC/SD卡、USB Host、USB OTG、VGA、S-Video、Keypad、WiFi、Led
wince驱动：NLED、GPIO/I2C/SPI/MCBSP、TV OUT、串口、6*6键盘、音频、Nand、LCD、VGA、TOUCH、SD/MMC/SDIO、DM9000网卡、WiFi、USB OTG、USB EHCI、VRFB、DSPLINKK/CMEMK、GPIO、PWM、ADC、ONENAND

- 硬件

原理图（pdf方式提供）
板载芯片数据手册

- 开发工具

linux交叉编译工具

- 开发文档

用户手册(包括硬件特性，linux用户指导及wince用户指导)

FAQ总结

关于使用SBC8100开发过程中所遇到的问题及解决方法，请访问SBC8100_FAQ

Links

- Chinese homepage (translate)

Translate

Translate this page to english

Categories:

- Linux

- OMAP
- Development Boards

> From: eLinux.org

SFFSDR

This page is for the Lyrtech **S**mall **F**orm **F**actor **S**oftware **D**efined **R**adio (SFFSDR) Board based on DaVinci DM6446 SoC from Texas Instruments.

- Lyrtech information about small form factor SDR development platforms
- Original page http://opensdr.com/node/6

Category:

- Development Boards

> From: eLinux.org

SheevaPlug

Contents

Introduction

From http://www.marvell.com/products/embedded_processors/developer/kirkwood/sheevaplug.jsp:

The SheevaPlug is a development platform, targeted for use as a plug computer, and designed to run network-based software services. It features a Kirkwood Series SoC with an embedded Marvell Sheeva™ CPU core running at 1.2 GHz. This device connects to the network using GbE, offers desktop class performance, and can be used to replace a PC-based home server for many applications. Peripherals connect using the included USB 2.0 port.

The development kit is enclosed in a plastic case that also contains a universal power supply. For developers a USB-based debug connection is included to enable simple debugging and reprogramming.

Hardware

- 1.2 Ghz ARM CPU (88F6281 aka kirkwood) (L1 Cache: 16K Instruction + 16K Data, L2 Cache: 256KB)
- 512 MB RAM
- 512 MB NAND flash
- 1 GBit ethernet
- USB host port
- Real Time Clock with batter
- JTAG and console interface via USB
- SDIO interface

Pictures of the internals can be found at http://www.cyrius.com/debian/kirkwood/sheevaplug/gallery.html

Software

Linux variants supported or in the works:

- Ubunto
- gentoo
- debian
- fedora
- openembedded

Purchasing and more info

You can order a sheevaplug for USD 99 from http://www.globalscaletechnologies.com/p-22-sheevaplug-dev-kit.aspx

Documentation can be found at
http://www.marvell.com/files/products/embedded_processors/developer/kirkwood/SheevaPlug_DocumentationPackage.zip

"Its hardware design is completely open -- everything from schematics to Gerber files will be available on a website ... volume pricing could fall to $50" [1].

Similar products

PogoPlug

http://www.pogoplug.com/

Specification is not clear, No JTAG/SDIO

Preorder: USD 79, retail price: USD 99.

highseclabs sheeva plug

http://www.highseclabs.com/sheeva_plug_sp1100.html

Only 128 MB RAM, no JTAG/SDIO

ctera CloudPlug

http://www.ctera.com/home/ctera-cloudplug.html

No further information available

Further reading

1. ↑ Linuxdevices.com "$100 Linux wall-wart launches"
2. http://www.plugcomputer.org/

Category:

- Development Boards

> From: eLinux.org

StalkerBoard

EMA OMAP3530 EV A2 EMA OMAP3530 EV A2评估板是广州英码信息科技有限公司自主研发的一款基于TIOMAP3530处理器的开发板。

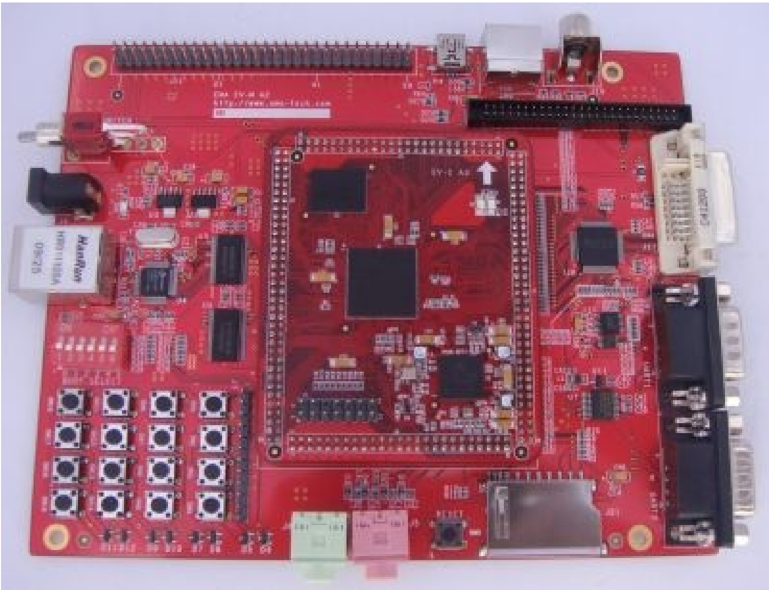
方便用户快速体验OMAP3530处理器的强大的数据运算处理能力，降低产品开发周期，实现面向消费电子、医疗仪器、多媒体处理、视频监控、工业控制等领域的产品快速上市。

该开发板为方案商、软件商、集成商提供了一个完善的OMAP3530评估/开发平台。

此工程的建立主要是为用户介绍EMA OMAP3530 EV A2的硬件软件资源及在开发OMAP3530时提供一个交流分享平台，可使用户快速的对EMA OMAP3530 EV A2评估套件进行全面的了解，并对开发的常见问题进行总结归纳。

Contents

评估板简介



EMA OMAP3530 EV A2评估主板

评估板概述

该开发板分为核心板与底板两部分。它的设计思路的核心是：以TI 最新的处理器OMAP3530为核心形成一个精简而完备的计算机系统。OMAP3530处理器基于性能四倍于AMR9内核的ARM Cortex-A8内核。OMAP3530在单一的芯片上集成了ARM Cortex-A8内核、DSP、图形引擎以及提供丰富的外设资源，提供了业界最佳的通用多媒体和图形处理单芯片组合。EMA OMAP3530 EV A2评估板采用OMAP3530作为其处理器，为开发者使用OMAP3530处理器提供了完善的软件开发平台，支持Linux及WinCE 6.0操作系统。OMAP3530提供超过 1200 Dhrystone MIPS 的高性能，可运行带桌面窗口管理器的 Linux 操作系统与办公应用，此外集成了兼容OpenGL ES 2.0的图形引擎，可针对游戏与 3D 用户界面加速实现照片般逼真的实时图形加速。时钟频率高达430 MHz 的DSP 内核为诸如家庭媒体中心、机器人系统、WEB 信息站以及数字指示牌、高清视频(720p)等应用提供强大的运算能力。底板则在核心板的基础上提供一个全接口的评估/开发平台。

评估板特点

核心板硬件特性

处理器	OMAP3530 主频600MHz 、CortexA8NEON SIMD协处理、430MHz TMS320C64x+ DSP协处理、POWERVR SG™ 2D/3D加速引擎。
存储器	1Gbit(128MByte) 低电压DDR @ 166MHz。
Flash	1Gbit(128MByte) 高速低电压SLC NandFlash。
JTAG	14Pin 1.8V JTAG。
指示灯	供电指示，电源管理模块指示。

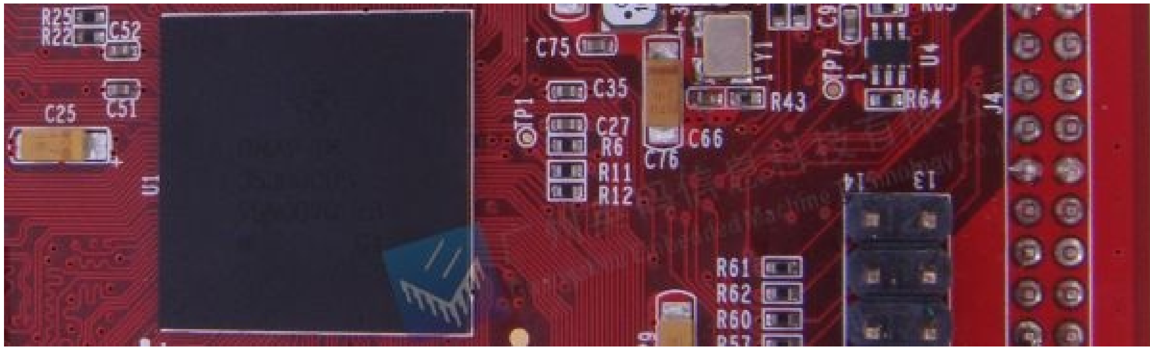
底板特性

电源插座	直流+5V电源输入。
电源开关	开关往下拨时，电源接通。开关往上拨时，电源关闭。
音频插座	立体声音频输入、立体声输出。
网络接口	标准RJ45 10/100M自适应以太网接口，带有链路灯、数据灯。
用户输入	复位按钮，一个4*4矩阵扫描键盘（最大支持到6*6）。
显示接口	1个数字视频DVI-D（支持EDID）、1个AV端口（复合视频接口）、1个SVIDEO OUT接口。
拨码开关	1个拨码开关设置启动设备顺序。
USB接口	高速USB 2.0接口，符合OTG（On The Go）、HOST两种标准。
串口通信口插座	2个标准的RS232型插座
SD/SDIO/SDHC插座	SD/SDIO/SDHC卡（也支持TF、MiniSD卡）
扩展接口	2*30插针扩展口 6*6矩阵扫描键盘扩展口 LCD扩展口

软件资源

系统	项目	特性	描述
Linux	启动代码	版本	x-load-1.42，U-Boot 2009.06-svn14
		启动方式	支持网络、SD卡或NAND Flash中启动引导Linux系统
		映像更新	支持网络或SD卡更新映像
	内核与驱动	版本	Linux 2.6.29
		支持文件系统格式	ROM/CRAM/EXT2/EXT3/FAT/NFS/JFFS2/UBIFS
		驱动支持	Serial, RTC, NET, NAND, LCD, Touch Screen, MMD/SD,USB OTG, DVI, Keypad
	文件系统	文件系统格式	Ramdisk文件系统、UBI文件系统
		系统特性	库支持（ALSA -lib, tslib, glibc）,udev设备管理支持
	Demo	Angstrom	音频（XMMS），网络（Firefox），图形编辑器（gimp）及文档处理软件(Abiword)
		Android	Google开发的基于Linux平台的开源手机操作系统
		DVSDK软件	支持MPEG4，MPEG2，H264，mp3，aac音\视频格式解码
WinCE	启动代码	版本	x-load-1.42、Eboot
		启动方式	支持网络、SD卡或NAND Flash中启动引导wince系统
		映像更新	支持网络或SD卡更新映像
	系统特性	内核特性	KITL内核调试, Reboot, Watchdog, RTC
		驱动支持	显示驱动（DVI, TFT LCD）
			SD卡, 键盘, McSPI, McBSP, 音频,网络, NLED, USB OTG
		系统功能	电源管理（背光驱动、电池驱动、休眠\唤醒功能）
			HIVE注册表支持
			ROM文件系统支持
		软件特性	Media play 9.0, Word编辑工具及Internet Explorer 6.0
			.NET Compact Framework 3.5

评估板、接口图片



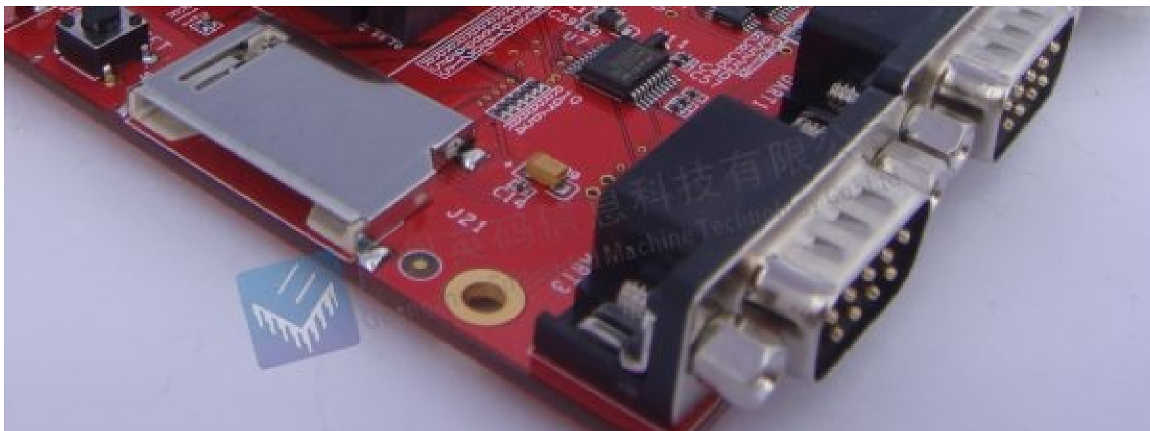
OMAP3530处理器

主频600MHz 、CortexA8 with NEON SIMD协处理、430MHz TMS320C64x+ DSP协处理、POWERVR SG™ 2D/3D加速引擎。



存储器+NandFlash

1Gbit(128MByte) 低电压DDR @ 166MHz，1Gbit(128MByte) 高速低电压SLC NandFlash。



SD卡接口 、UART 串行口

可接入标准SD/SDIO/SDHC卡（也支持TF、MiniSD卡）卡。系统可从此插座插入的MMC/SD 卡启动。 2 路 UART 串行口，波特率可高达 115200bps，并具有 RS232 电平转换电路。



100M 以太网接口、拨码开关

标准RJ45 10/100M自适应以太网接口，带有链路灯、数据灯。



音频输入、音频输出和复位按钮

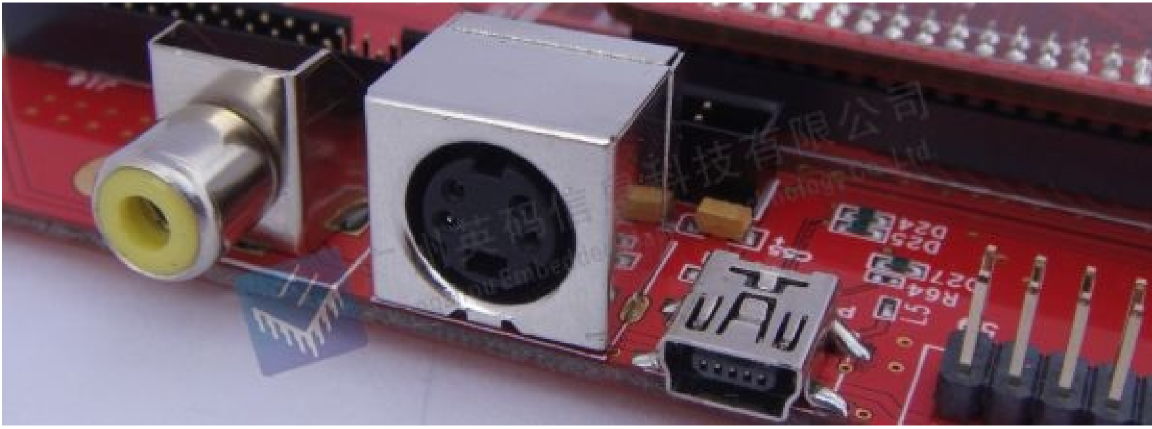
支持音频输入和音频输出，音频模块由 S3C2440 的 IIS 音频总线接口和 UDA1341 音频编码解码器组成。



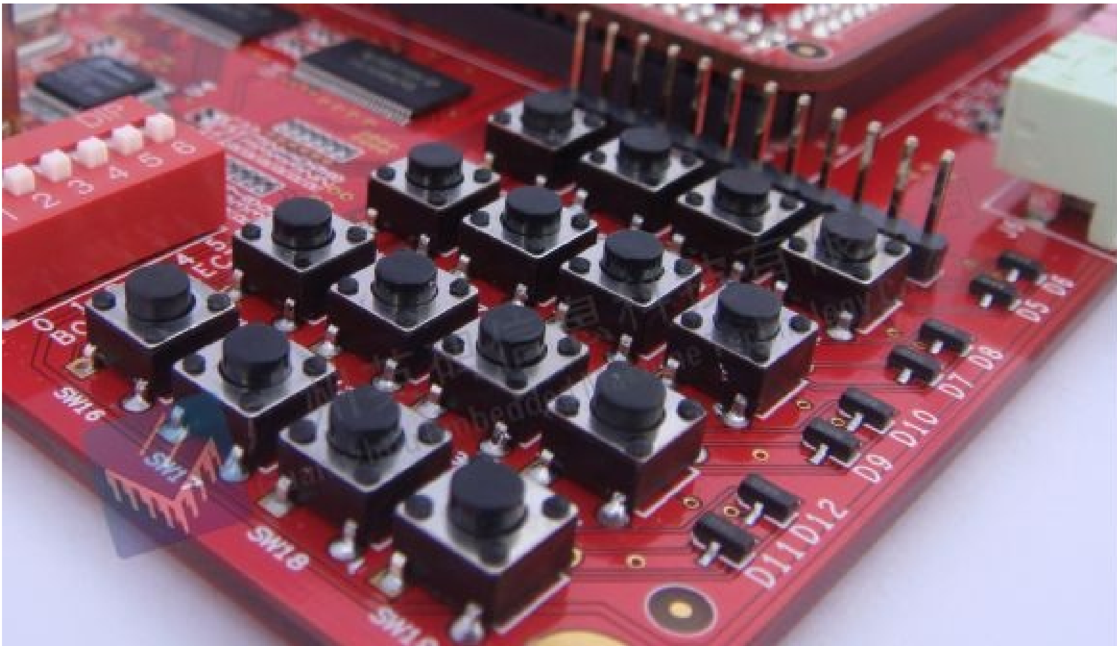
数字视频DVI

EMA OMAP3530 EV A2评估板可通过此端口驱动一个DVI-D输入接口液晶显示器。

通过此座，该板输出24位的数字彩色显示信号。此接口提供的I2C通讯了实现对显示器的识别与通信。



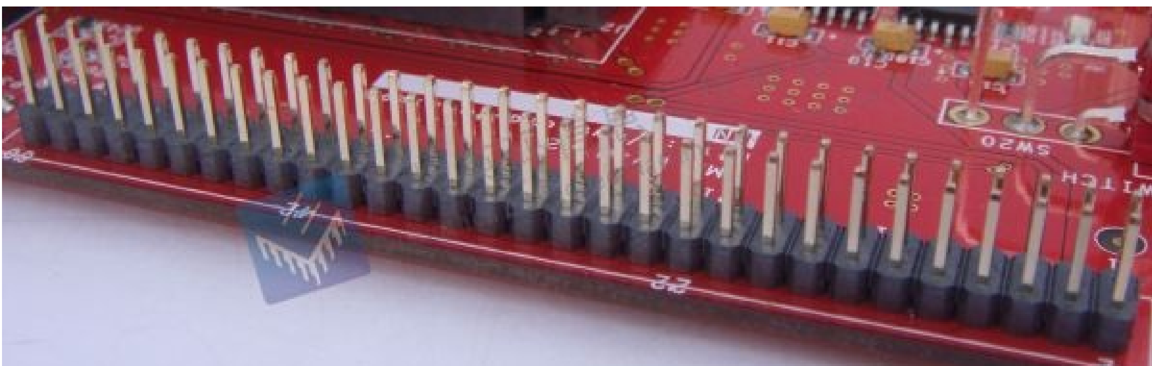
AV端子（CVBS）、SVIDEO OUT 插座、MiniUSB接口



4*4矩阵扫描键盘



LCD接口插座如图



扩展外设插座

评估板接口详述

（1）+5V直流电源输入插座J9

J9为+5V直流电源输入座，注意输入电源插头的中芯为DC+5V，插头外壳为直流电源地，输入电源最小800mA（推荐为1A）。

（2）电源开关插座SW20

当往下拨时，电源处于接通状态。往上拨时，电源处于关闭。

（3）网络接口插座J10

此接口为标准RJ45 10/100M自适应以太网接口，带有链路灯、数据灯。

（4）拨码开关J12（备注：1： 开关向上拨 0：开关向下拨）

通过拨码开关设置OMAP3530启动设备的顺序。

（5）耳机输出插座J6

提供立体声（双声道）音频信号输出，请在此端口连接有源音箱的音频输入插头或者立体声耳机/耳塞。

（6）麦克风输入插座J5

提供立体声（双声道）音频信号的输入，请在此端口连接标准的立体声麦克风音频输入插头。

（7）复位开关SW5

当按下并释放时，会使EMA OMAP3530 EV A2评估板进入上电复位状态，可以使开发板重新启动。

（8）SD卡插座J21

可接标准SD/SDIO/SDHC卡。
系统可以从此插座插入SD/SDIO卡启动。

（9）串口通信插座UART1 、UART3

串口插座提供标准的RS232通讯信号到其他设备。使用标准串口（直通）电缆连接开发板和其他标准串口接口的设备（如通用的PC机）。UART3是Linux与WinCE系统的默认TTY。

（10）数字视频DVI插座J1

EMA OMAP3530 EV A2评估板可通过此端口连接一个DVI-D输入接口液晶显示器。通过此接口，标准TMDS数字彩色显示信号。此接口提供的I2C通讯了实现对显示器的识别与通信。

在连接评估板和显示器的DVI-D接口时，要选用DVI-D型视频连接线。在选配液晶显示器时，除了要兼容DVI-D接口，还建议选用能够支持1280*720或以上分辨率的显示器。

（11）液晶接口插座

该接口包含24位真彩色CMOS显示信号与触摸屏接口，所接液晶屏显示的分辨率可达到1920*1080。

（12）AV端子（CVBS）

复合视频端子也叫AV端子或者Video端子，是目前最普遍的一种视屏接口。EMA OMAP3530 EV A2评估板选用AV接口，用来连接混合视频信号，为黄色插座。

（13）SVIDEO OUT 插座 J20

此插座可用视频线连接标准电视设备，实现复合视频信号的输出。EMA OMAP3530 EV A2评估板可同时分别输出此复合信号和DVI-D信号，从而实现不同的显示输出。

（14）MiniUSB接口P3

此接口为Mini 型USB 主从复用插座，可连接标准的USB从设备。支持OTG、HOST两种模式（配用不同标准的接线）。
当需要将此接口当作从口使用时，可选择标准MiniUSB 转USB 接口电缆。当需要将此口当作主口使用时，请使用MiniUSB 转USB A转接插件，再连接一只有源的USB 集线器使用。

（15）4*4矩阵扫描键盘

列\行	0	1	2	3
3	SW16	SW11	SW6	SW1
2	SW17	SW12	SW7	SW2
1	SW18	SW13	SW8	SW3
0	SW19	SW14	SW9	SW4

（16）扩展外设插座J11

扩展插座提供部分可外接的扩展信号，用户在需要使用这些信号时，可用相应的插座取得这些信号来利用。
请注意：J11中VBAT提供4.2V的输出。其他提供的信号均为1.8V标准，用户在连接信号的时候，需要注意其电平限制，否之容易导致评估板上芯片损坏。

(17) 电源指示灯

当EVA OMAP3530 EV A2评估板，正确输入DC+5V电源后，此指示灯会亮起来，提示供电正常。如果供电后指示灯不亮，请立即断开电源并查找故障原因。

评估板特性

提供了完善的软件开发环境，支持Linux操作系统及WinCE 6.0操作系统

性能强劲：OMAP3530处理器采用600MHz主频的ARM Cortex-A8内核，配合NEON technology，可满足大部分中高端多媒体应用需求 内置DSP：内置430MHz的TMS320C64x+ DSP核，可支持H.264的硬件编码/解码加速 2D/3D硬件加速：内置POWERVR SG™ 2D/3D图形硬件加速引擎 低功耗：核心板在全速情况下不超过2.5W功耗 采用最小系统分离式的设计：该产品的核心板提供了一个基于TI OMAP3530处理器的最小系统。整块核心板体积细小但功能完善, 并提供OMAP3530处理器的全部接口。根据不同的客户需要，只需要替换底板即可满足不同的应用场合。

硬件研发难度低、成本低：硬件可基于核心板直接进行研发，一般情况下只需要设置个简单的双层电路板设计即可完成硬件研发。

方便用户快速体验OMAP3530处理器的强大的数据运算处理能力，降低产品开发周期，实现面向消费电子、医疗仪器、多媒体处理、视频监控、工业控制等领域的产品快速上市。

应用领域

低功耗平板电脑：使用核心板的基础上设计一块简单的底板，再搭配一块8寸LCD即可满足应用要求。OMAP3530可支持Linux、Windows CE系统，内建600MHz ARM Cortex-A8内核的处理能力完全满足低功耗平板电脑的要求。

高性能手持设备：使用核心板的基础上设计一块简单的底板，再搭配一块小尺寸LCD即可满足应用要求。低功耗的核心板设计提供更高的设备续航能力。例如智能手机、GPS系统。

IP视频监控终端：使用核心板的基础上设计一块简单的底板，再搭配一个监控摄像头即可满足应用要求。OMAP3530内置的TMS320C64x+ DSP核可进行硬件H.264编码，使得监控视频流在最小带宽占用下稳定传输。

大型游戏机：使用核心板的基础上设计一块简单的底板，再搭配各种大型游戏机的输入设备即可满足应用要求。OMAP3530内置的POWERVR SG™ 2D/3D图形硬件加速引擎可以完全满足大型游戏的3D性能要求。

医疗仪器：使用核心板的基础上设计一块简单的底板，通过VGA显示即可满足应用要求。

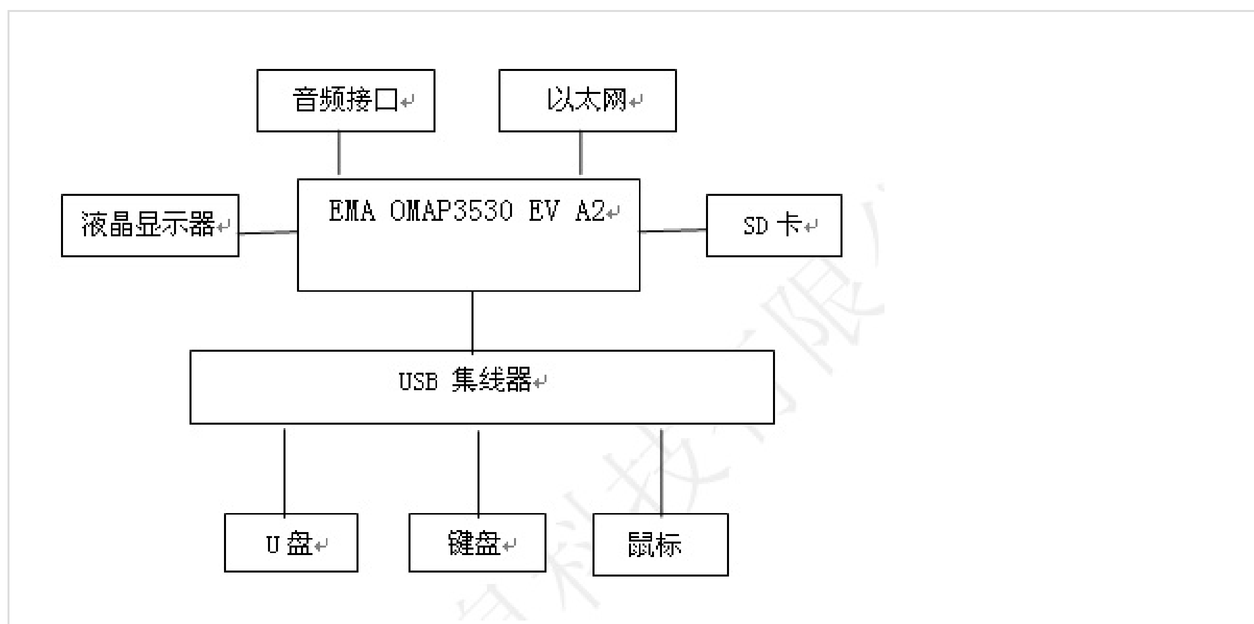
工业控制：使用核心板的基础上设计一块简单的底板，再搭配工业用的、CAN总线、网口等接口即可满足应用要求。OMAP3530内置的430MHz的TMS320C64x+ DSP核，可满足工业控制的应用开发要求。

核心板硬件资源特性

EMA OMAP3530 EV A2核心板硬件配置

OMAP 3530	600MHz ARM Cortex-A8 processor with NEON technology
	430MHz TMS320C64x+ DSP协处理器
	POWERVR SG™ 2D/3D加速引擎
RAM	1Gbit(128MByte)/ 2Gbit(256MByte) 低电压DDR @ 166MHz
Flash	1Gbit(128MByte)/ 2Gbit(256MByte) 高速低电压SLC NandFlash
指示灯	供电指示，电源管理模块指示
JTAG	14Pin 1.8V JTAG
电源输入	4.2V DC输入
电源输出	1.8V DC输出（配给其他接口器件用）
I/O电压	1.8V LVCMOS

以评估板为核心的计算机系统



评估板系统配件

标准配件

1、 EMA OMAP3530 EV A2评估板一块 2、 直流+5v的电源一只 3、 RS232串口线1条。直通线，两端均为母头。 4、 2G SD卡一块 5、 USB2.0 Mini B 转USB2.0 A 线 6、 软件光盘1张 DVD

扩展配件

1、 USB 键盘 2、 USB鼠标 3、 音频连接线（双声道）1条 4、 4口USB集成器 5、 网线

nandflash 启动教程

EMA OMAP3530 EV A2评估板已成功从nandflash 启动。更详细的教程，请参考http://code.google.com/p/ema3530/wiki/NandBooting

光盘内容

OMAP3530开发板用户手册

OMAP3530核心板用户手册 OMAP35X技术参考手册

OMAP3530数据手册

OMAP3530底板A2版芯片数据手册

Linux镜像文件(MLO、u-boot.bin、uImage) X-Loader二进制程序 U-Boot二进制程序 Linux内核镜像文件 Demo文件系统 X-Loader 源代码 U-Boot 源代码 Linux kernel 源代码 Linux 交叉编译工具链 Ubuntu虚拟机系统压缩版

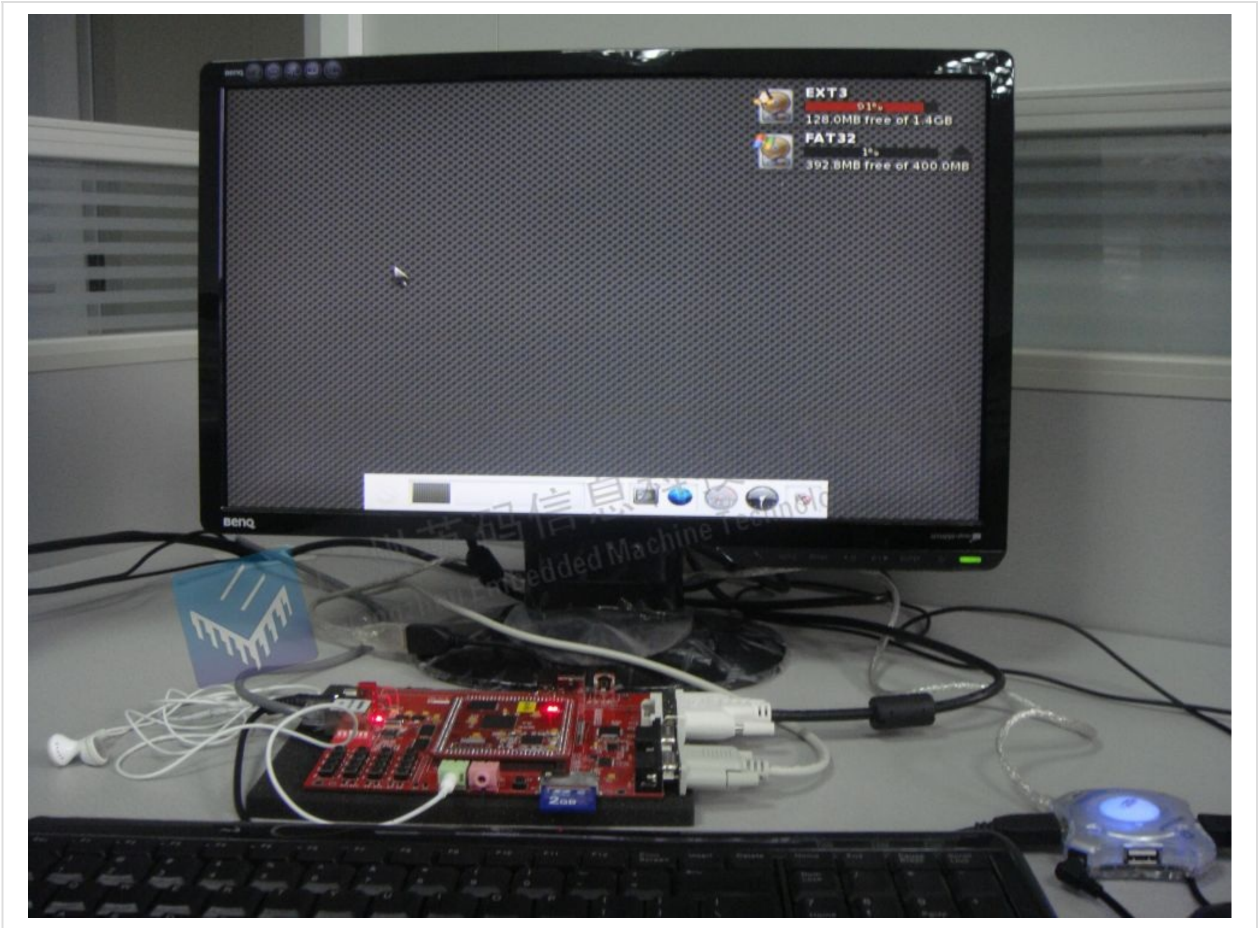
EMA OMAP3530核心板A3版管脚原理图 EMA OMAP3530底板A2版原理图

Tftpd下载工具 虚拟机安装包

Demo系统运行效果

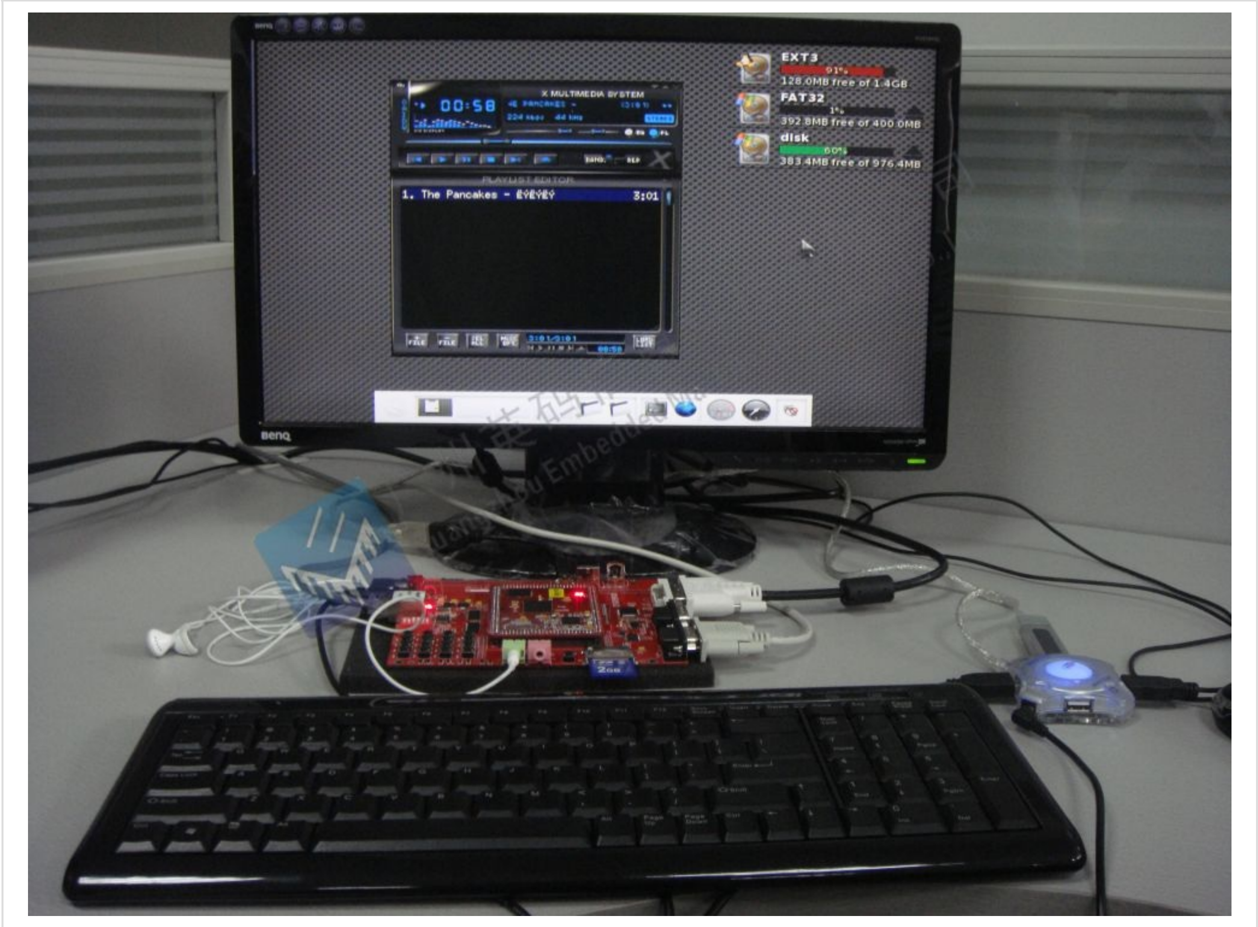
本系统提供的Demo程序为基于Angstrom的Linux桌面系统。 透过DVI-D信号输出的界面(分辨可达1080p)。 可使用Angstrom系统中的各种软件，包括文档编辑，上网浏览，音频视频播放及图形编辑等功能，并可用USB键盘鼠标对此系统进行操作控制。




EXT3
128.0MB free of 1.4GB
FAT32
392.8MB free of 400.0MB
BenQ

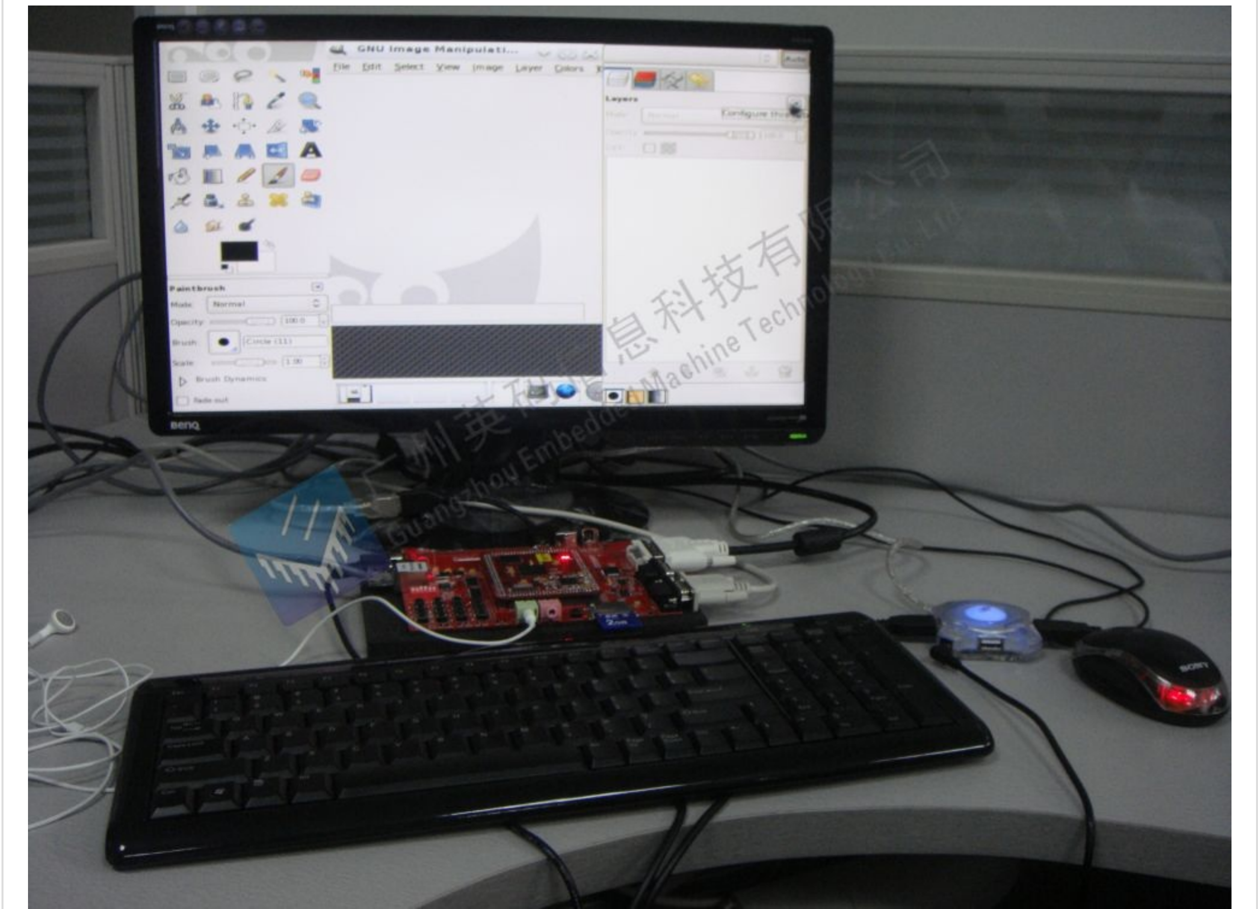

EXT3
128.0MB free of 1.4GB
FAT32
392.8MB free of 400.0MB
Main
Applications
Files
Run Command
Desktop
Windows
Lost Windows
Enlightenment
Settings
System
Accessories
Games
Graphics
Internet
Multimedia
Office
Settings
Connection Manager
Firefox Browser
Midori (Web Browser)
Pidgin Internet Messenger (Internet Messenger)
BenQ


Untitled1
File Edit View Insert Format Tools Table Documents Help
Page Width
Normal
Times New Roman
12
Page 1/1
INS
default
en-US

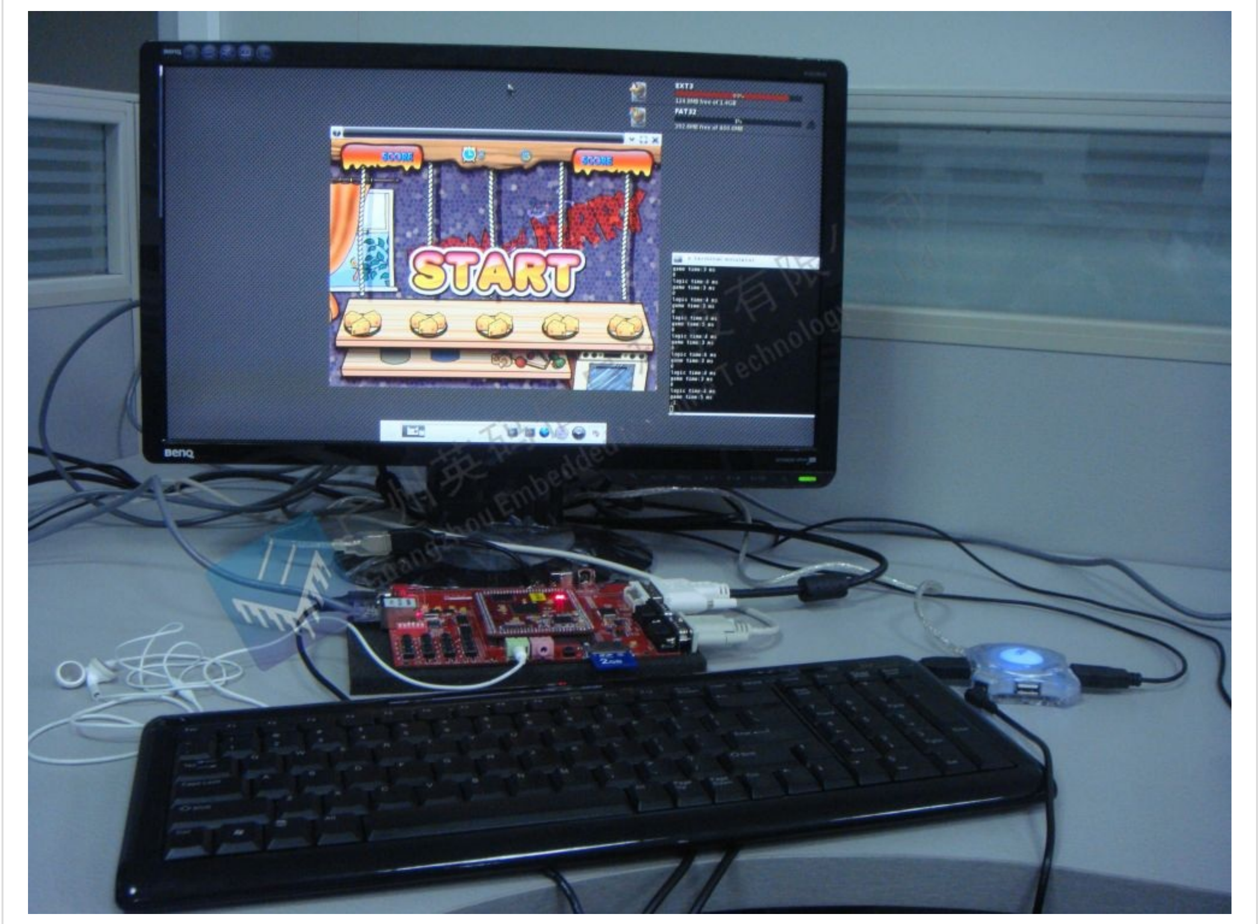

EXT3
128.0MB free of 1.4GB
FAT32
392.8MB free of 400.0MB
disk
383.4MB free of 976.4MB
00:58
X MULTIMEDIA SYSTEM
1. The Pancakes
3:01



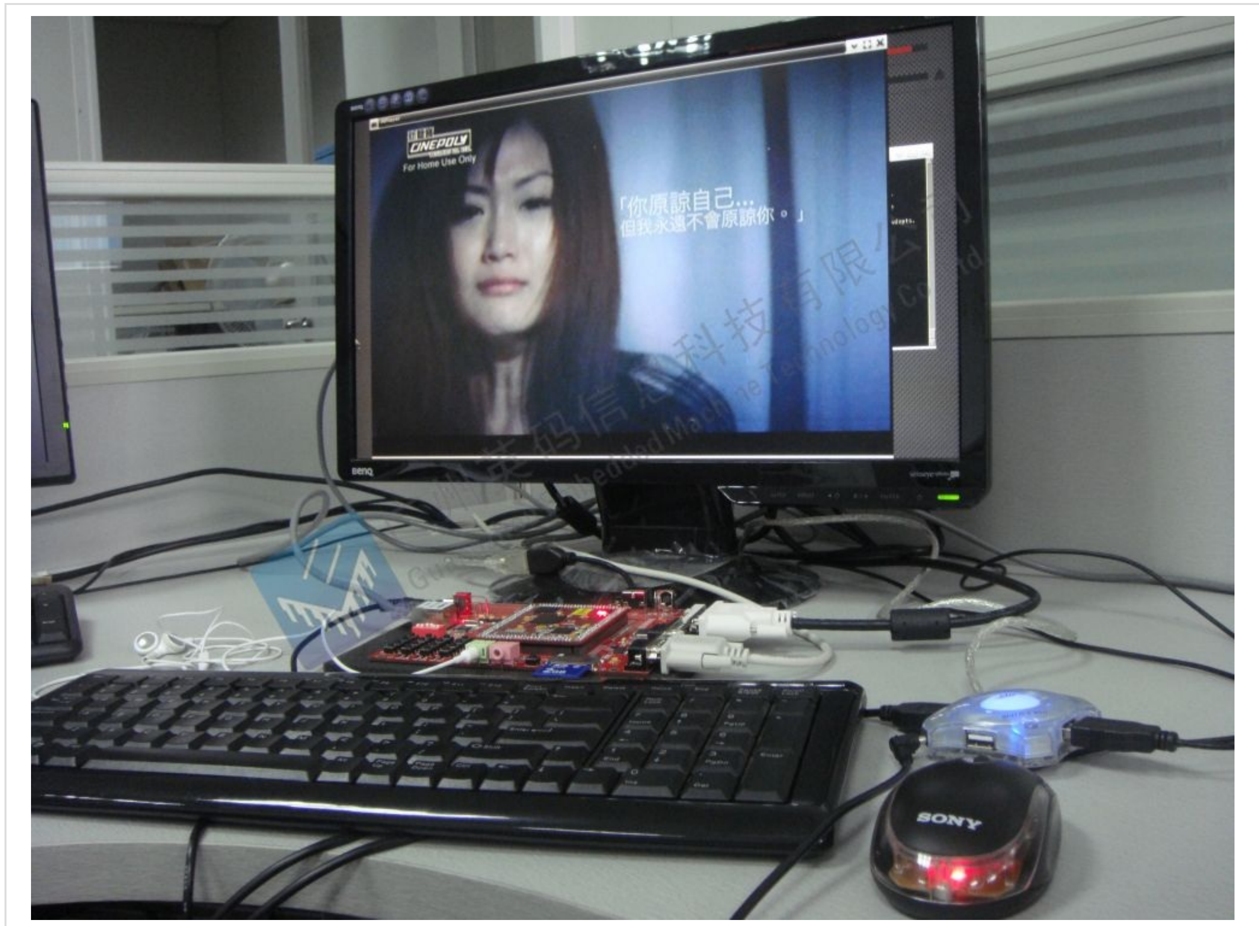
File Edit View History Bookmarks Tools Help
Most Visited
Getting Started
Latest Headlines
TEXAS INSTRUMENTS
Products
Applications
Design Support
Sample & Buy
OMAP3530 Status: ACTIVE
Applications Processor
View RoHS Compliant Devices
Most Popular Links
Samples
Pricing/Packaging
Technical Documents
Features
Related Products
Tools & Software
Inventory
Waiting for focus.ti.com...



GNU Image Manipulati...
File Edit Select View Image Layer Colors
Layers
Paintbrush
Mode
Normal


START


Vase
Translucency and reflections.
SONY



Categories:

- Linux
- OMAP
- Development Boards

> From: eLinux.org

TechnologicSystems

Technologic Systems

Technologic Systems (http://www.embeddedarm.com) offers some nice boards that are usable as development boards but also as OEM modules to integrate into your products.

Especially the TS-7300 is a very nice board build around the Cirrus Logic EP9312 SoC to experiment with FPGA technology. This board contains a completely unoccupied FPGA that can be programmed from the main CPU.

Category:

- Companies

> From: eLinux.org

Tegra2

Tegra 2 is a system-on-a-chip (SoC) created by NVIDIA. The SoC comprises a dual-core ARM Cortex-A9 with FPU per core, an NVIDIA ultra low power (ULP) GPU, an embedded ARM7TDMI and five other processing cores for audio and video processing.

The NVIDIA developer zone has more information on Tegra.

There are forums for software developers and the like.

Topics

- Hardware
  - Tegra 2 Chip
  - Tegra 200/250 Development Board
  - Colibri T20 Computer Module
  - Devices
    - Rumored
    - Announced
- Operating Systems
  - Android
  - Linux
  - Debian
  - Windows RT
  - Windows CE, Windows Compact
- OpenGL Information and Development

Hardware

[TODO: Add information about dev kit hardware]

Category:

- ARM processors

> From: eLinux.org

Tegra/Mainline SW/U-Boot

\< Tegra | Mainline SW

NVIDIA Tegra SoCs are well supported by mainline U-Boot.

U-Boot releases may be obtained from:

- ftp://ftp.denx.de/pub/u-boot/ releases, via download.
- http://git.denx.de/?p=u-boot.git;a=summary releases and overall development code, via git.
- http://git.denx.de/?p=u-boot/u-boot-tegra.git;a=summary Tegra-specific latest development code, via git.

Features that already work are:

- It boots!
- Serial console.
- SD/eMMC.
- USB2 Host:
  - USB networking for network booting.
  - USB keyboard enabled on some boards.
- Display controller, for some SoCs and boards - mainly Tegra20/30.
- `bootz` , `sysboot` , `pxe` commands, shell, and scripting available for easy distro booting support.
- Extremely basic PMIC support necessary to boot the main CPUs.
- I2C master.
- SPI master (e.g. for boot flash programming).
- NAND (Tegra20 only, for boot flash programming).
- I2C slave (Toshiba AC100 specific NVEC keyboard support).

A probably-incomplete list of features that are not yet implemented is:

- Advanced (high-speed) transfer modes for SD and eMMC. IO voltage scaling.
- PCIe (e.g. for PCIe-based network devices on some boards).
- USB2 device mode and OTG. (device mode support is work-in-progress, and nearly complete).
- USB3 controller.
- SATA.
- SKU awareness (SKU-specific clock and thermal limits).
- POR (Plan Of Record) clocks.
- HDMI display.
- LCD panel support on many boards (especially Tegra114 and later).

Categories:

- NVIDIA
- Tegra
- U-Boot

> From: eLinux.org

Tiny210

- Mini210S Dark colored Release
- Mini210S African american Release
    - -- 1GHz Central processing unit
    - 1GB SLC / 512MB Ram memory
    - 2.0 Or Android operating system

Apart from the masterdom to become dark colored, and also declaring "ARMWorks", it possesses a great few additional features:

Make of this description what you will...

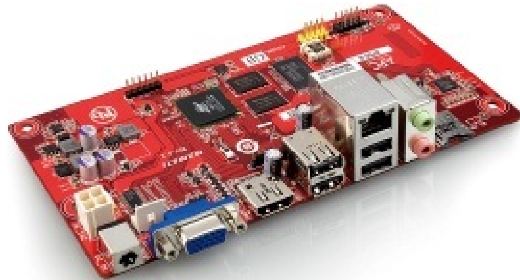
1st, this micro, ADC analyze cooking pot, as well as the piezo buzzer can easily all be eradicated (unsoldered : they're by ditch parts) plus a 6 flag outlet provides admission to those signs. You can see the idea while in the decrease still left listed here, from the customer switches. OEMs might sequence planks without worrying about a couple of sections and also the GPIO as well as coach bus extendable headers unpopulated. Oh yea, and 1G SLC NAND to make certain you will find free owners.

Links

- http://www.china-pbx.cn/friendlyarm/Tiny210/
- http://www.voice-logger.com/arm/Tiny210/

> From: eLinux.org

VIA APC 8750



Introduction

APC runs a custom Android system, built for keyboard and mouse input. A basic selection of applications is preinstalled. Also included is a full set of consumer I/O ports, enabling APC to connect to your PC monitor or TV.

More info on http://apc.io/

Specifications

Item	Description
Model	APC 8750
Software	Android 2.3 (PC System)
Chip	VIA 800MHz Processor
Memory	DDR3 512MB Memory 2GB NAND Flash
Graphics	Built-in 2D/3D Graphic Resolution up to 720p
Input and Output	HDMI / VGA / USB 2.0 (x4) / Audio out / Mic in microSD Slot
Network	10/100 Ethernet
Size	170 x 85mm (W x H) Neo-ITX Standard*

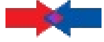
* Neo-ITX is form factor compatible with Mini-ITX and MicroATX

Category:

- VIA APC

> From: eLinux.org

WandBoard

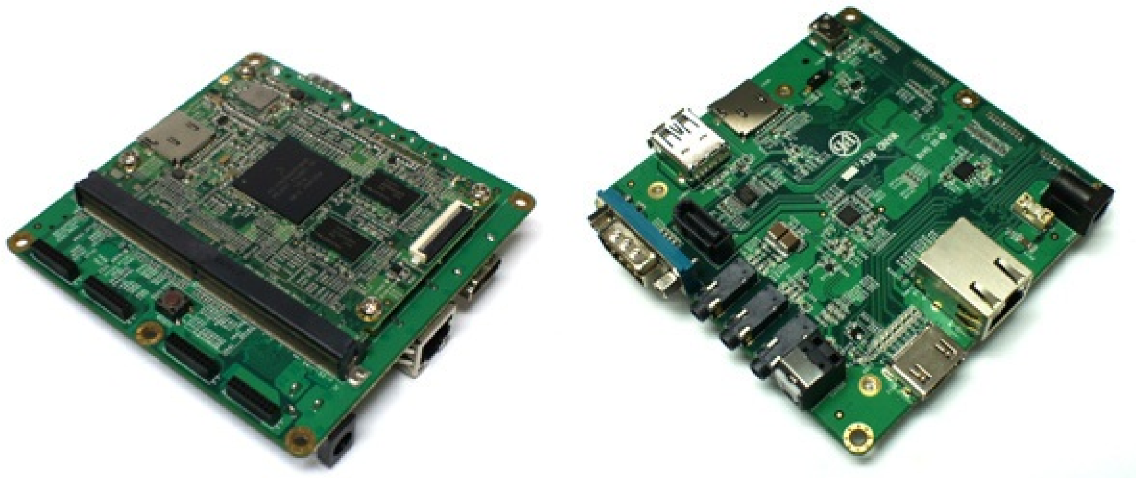
	This page is may need to be merged with other page(s) including Wandboard. Please help to merge the articles, or discuss the issue on the talk page.

The [WandBoard] is a very low-cost development platform based on the Freescale i.MX6 processor.

Contents

Hardware

There are currently two versions of WandBoard, "Solo" and "Dual". The WandBoard itself is a System-On-Module of the EDM standard that comes with a matching motherboard. The motherboard is quite trivial and is the same for both versions of the WandBoard.



EDM module

- Freescale i.MX6 CPU (ARM Cortex A9)
  - i.MX6 Solo (single core) on "Solo" version
  - i.MX6 Dual Lite (dual core) on "Dual" version
  - Vivante GPU:
    - 3D core: GC880 @ @ 533MHz, 1 shader core 4.8 GFLOPS
    - 2D core: GC320 composition engine
- DDR3 RAM
  - 512 MB on "Solo" version (two 2Gbit chips)
  - 1 Gigabyte on "Dual" version (four 2Gbit chips)
- MicroSD card slot (bootable, at SD3 of the CPU)

- AR8031 Gigabit LAN
- BCM4329 Wi-Fi+Bluetooth (at SD2, UART3, AUD5; unpopulated on "Solo" version)
- Flat cable connector for camera (CSI, I2C2, GPIO)
- Power circuit (5V input)

Motherboard

- Gigabit LAN connector
- HDMI connector
- USB host connector (USB3 for future use but currently in USB2 mode only)
- USB OTG connector
- MicroSD card slot (at SD1)
- UART DB9 connector (at UART1 via ADM3202)
- Audio I/O (at AUD3 via SGTL5000) and SPDIF (3.5mm jacks and optical)
- 5VDC power barrel connector
- SATA connector (probably unpopulated, reserved for future use; no SATA on i.MX6 Solo and Dual Lite)
- Four 1.27mm 2x10 expansion headers:
  - LVDS0 (4 signal pairs + clock), +3V3
  - Flat panel display data lines (24 bit total); no power or GND at this header
  - Rest of FPD data lines, FPD control lines, +3V3, three I2C interfaces
  - One SPI (with two CS), 8 GPIO lines, +5V

Availability

As of Jan 17, 2013 preorder only. Prices are 69 USD for Solo and 89 USD for Dual. First boards seem to be shipped to partner developers.

Community

- Website: http://wandboard.org

Categories:

- NeedsMerge
- Development Boards
- Hardware
- SBC
- ARM

> From: eLinux.org

Kernel Mainlining

This page has information for embedded developers about mainlining patches to the Linux kernel.

Contents

General Resources

- Documentation/HOWTO
  - kernel document describing how to code for the kernel and contribute
- Documentation/development-process
  - kernel documents describing the kernel development process

Presentations

Greg KH has a great presentation about how the community works, with links to references for getting started:

- Linux Kernel Development (pdf)

An older talk (2008) by Andrew Morton discusses the reasons to contribute, and best practices for contributing to the upstream kernel

- Session:kernel.org development and the embedded world
  - In this seminal talk from 2008, Andrew lays out the case for involvement of embedded companies in kernel development. He describes the overall process, but more importantly what to expect from kernel developers, what to do and not to do when approaching mainlining, and how to structure teams for effective work with the kernel community.

talk list

Here is a list of talks about mainlining and community involvement, from previous Linux conferences:

- How to Participate in the Kernel Development Process (PDF)
  - ELC-2007, April 2007, Jonathan Corbet
  - This talk is an attempt to identify the factors which lead to success or failure and present them in a way that will help others seeking to get code into the kernel.
- ELC-2008 morton (noted above)
- Appropriate Community Practices: Social and Technical Advice - ELC-2008, April 2008 Deepak Saxena

  - Abstract: With the increasing popularity of Linux in the embedded world, HW vendors are jumping on the bandwagon to add kernel support for their devices/chipsets/SOCs. We in the community keep seeing the same mistakes made (both technical and social) repetitively. We will go over the benefits of being involved with the community and utilize examples of what not do when working within the Linux development ecosystem to illustrate appropriate practices to increase your probability of successful code adoption into the kernel.org tree.
  - (presentation not available)
- Embedded maintainers: Community and Embedded Linux ELCE-2008, David Woodhouse
  - This presentation introduces and discusses the new community rôle of 'embedded maintainer', present David's ideas and seek other opinions on what the job is actually supposed to mean.
  - The community at large needs to be more coherent - it's not just about big companies playing nicely with us, but also about building a community around embedded Linux in a way that we haven't really done so far. Even the individual projects aren't working together as well as they should. The 'embedded maintainer' rôle isn't like other maintainers in the kernel - we don't own a certain section of the code and just act as gatekeeper and arbiter of taste for it. It's more about bringing people together and getting them to collaborate better.
- Embedded Linux and Mainline Kernel
  - ELC-2009, April 2009, David Woodhouse
  - Embedded Linux has more in common, technologically, with other Linux use areas than many embedded developers realize. In this talk, David will describe some of the important intersections between the features embedded developers care about and those needed for enterprise and desktop systems. The stereotype of embedded developers not needing to interact with the greater Linux community is wrong. David provides the technical rationale for increased interaction in the community as well as tips for better involvement by embedded developers.
  - Notes: find other parties with same requirements - look outside embedded. Virtualized systems is a good place to look, as they often have resource constraints as well.
- Cooperative Development Inside Communities ELC-2009 Jeff Osier-Mixon
  - This is the talk introducing MELD.
- Becoming Part of the Linux Kernel Community ELC-2011, April, 2011, Arnd Bergmann
  - This talk give the benefits of beign integrated with the community (it was the "hippy" talk)
- Developer's Diary: Helping the Process ELC-2011, April 2011, Wolfram Sang
  - include notes on best practices for contributing to mainline
- Contributing to the Community? Does your manager support you? - ELCE-2011 Satoru Ueda
  - This is a "how to convince your manager" talk.
- ELC-2013 rose
- ELC-2013 chalmers
- ELC-2014 maupin
- Two years of ARM SoC support Mainlining: Lessons Learned ELC-2014, April 2014, Thomas Petazzoni
  - Give many good tips, including social ones

Training, tutorials and challenges

- The KernelNewbies web site is specifically dedicated to helping developers learn how to make contributions to the Linux kernel
  - There's a "To Do" list of small tasks that are ready for development, here http://kernelnewbies.org/KernelJanitors/Todo
  - See their excellent Guide to Upstream Merging for a lot of good technical and social tips.
- The Outreach Program For Women has an excellent tutorial on the steps for contributing one's first patch to the kernel
  - OPFW First Patch tutorial
- Eudyptula Challenge
  - This is a series of 20 tasks, managed via e-mail, that help people learn how to work with the kernel and submit patches
  - LWN.net article http://lwn.net/Articles/599231/
- http://www.tuxradar.com/content/newbies-guide-hacking-linux-kernel

Specific Projects

- CE Workgroup Device Mainlining Project
- Qualcomm SOC Mainlining Project
- Allwinner mainlining effort
  - this is a really good page with a write-up of the status of mainlined items for each kernel, and what tasks remain

Notes for Best Practices

from Andrew Morton

- Industry should have an embedded maintainer
- Report problems and requirements upstream
- Participate in community forums
- Companies should dedicate a few developers separate from product teams
- Develop product on latest mainline kernel, freeze it at end of product development
  - My aside: Current nature of Android features and board support preclude this
- Ask the community (Andrew) for help

from Deepak Saxena

- don't be arrogant - don't assume your experience in proprietary development methods translates into open source
  - be humble and listen to others
- release early and often
  - not doing this wastes a lot of time on implementations that get discarded, rewritten
- do your homework
  - see what's already there in Linux, and whether it can be extended to support your case
  - add to existing abstractions rather than add your uniqe solution (be willing to abandon your code, as long as you ultimately get support for your feature upstream)
- don't add OS abstractions (or, HALS for other OSes)
  - drivers must be Linux native - other layers and abstractions complicate the drivers - they can't be maintained by Linux kernel developers
- do add abstractions - don't just solve your immediate problem
  - write systems that suport multiple related hardware
  - be willing to generalize
- do your homework
  - use mainlining resources
  - ask informed questions
- work with the community - treat them as equals on your team
  - treat external developers input as you would your own team members
  - be respectful

from Jonathan Corbet

- why - \
- difference between proprietary and open source software
  - proprietary = product-driven, top-down requirements, short-term, internal QA, hierarchical decisions, private code base, complete control
  - open source = process-drive, bottom-up requirements, long-term, external QA, consensus decisions, public code, little control
- Understand the patch life-cycle
  - post early, fix things with community

    - get to staging
    - acceptance into mainline
- post early and often
- submitting patches
    - send changes - can influence direction even if not accepted
    - no: multi-purpose patches - make each patch small and indpendent
    - make patch serieses bisectable
    - follow submission rules
        - use diff -u, no MIME, correct format, Signed-off-by line, watch word-wrapping
    - send to correct place: MAINTAINERS, get-maintainer.pl
    - listen to reviewers, be polite, don't ignore feedback
- let go
    - your code may be re-written or replaced
- Coding
    - follow the style guides
        - not too much (HAL layers, unused parameters, single-line functions)
            - no multi-OS code
        - not too little - should generalize if there's already existing code
    - don't break APIs
        - can break internal APIs (only with very good reason), but you must fix all in-tree code
        - NEVER, NEVER break user-space API
    - don't cause regressions

from Arnd Bergmann

- Friend, Fans and Freeloaders
- don't annoy your kernel maintainer
    - publish all your code, including device drivers
    - would really like open source 3d embedded graphics drivers
- Being part of the community
- Give and Take
    - Divide and Conquer
        - Use public source code
        - break up source code - make a git branch for each feature
        - each branch should chance of getting upstream
    - Riding the Wave
        - all should be re-based as often as possible
    - Separate product and development trees
        - keep development in separate branches
    - Review
        - provides learning experience
        - newcomers can review and learn in the process as well
    - Respect
        - reviewers - should acknowledge effort of people working hard, even if you have to reject their stuff
        - submitters - should respect experience and knowledge of reviewers - follow their advice even if you don't agree with it
    - Rejection
        - maintainers - rejecting bad code is more important than accepting good code
    - Responsibility
        - don't duplicate infrastructure - extend it, generalize it

From David Arlie

http://airlied.livejournal.com/80112.html

---

you have a long road to walk, but first you have to leave the house

or why publishing code is STEP ZERO.

If you've been developing code internally for a kernel contribution, you've probably got a lot of reasons not to default to working in the open from the start, you probably don't work for Red Hat or other companies with default to open policies, or perhaps you are scared of the scary kernel community, and want to present a polished gem.

If your company is a pain with legal reviews etc, you have probably spent/wasted months of engineering time on internal reviews and stuff, so think all of this matters later, because why wouldn't it, you just spent (wasted) a lot of time on it, so it must matter.

So you have your polished codebase, why wouldn't those kernel maintainers love to merge it.

Then you publish the source code.

Oh, look you just left your house. The merging of your code is many many miles distant and you just started walking that road, just now, not when you started writing it, not when you started legal review, not when you rewrote it internally the 4th time. You just did it this moment.

You might have to rewrite it externally 6 times, you might never get it merged, it might be something your competitors are also working on, and the kernel maintainers would rather you cooperated with people your management would lose their minds over, that is the kernel development process.

step zero: publish the code. leave the house.

(lately I've been seeing this problem more and more, so I decided to write it up, and it really isn't directed at anyone in particular, I think a lot of vendors are guilty of this).

\< article is about why you should publish code immediately >

- This raises these issues:
  - why publish early? - because not doing so is possibly twice as much work
  - what barriers are there to publishing immediately:
    - dependencies!!! (mostly version gap)

Overcoming Obstacles to Mainlining

Tim Bird has prepared a talk about Overcoming Obstacles to Mainlining for ELCE 2014. See that page for information about this presentation and a link to his slides.

Category:

- Kernel

> From: eLinux.org

CE Workgroup Device Mainlining Project

This page describes the "Device Mainlining" project of the CE Workgroup.

This purpose of this project is to decrease the amount of out-of-mainline patches for the Linux kernel, in modern consumer electronics products.

This project was proposed at the CE Workgroup Steering Committee meeting in Tokyo, Japan, in June of 2014.

Contents

Activities

In terms of overall strategy, the actions being performed for this project fall into 4 categories:

- communicate and meet to make plans and carry forth objectives
- identify problems
- implement solutions
- train and educate participants

Communicate and meet to make plans and carry forth objectives

- see meetings section below

Identify problems

- Survey
  - Conduct a survey to identify obstacles to mainlining - Done, September 2014
- Present results from survey:
  - Presented "Overcoming Obstacles to Mainlining" talk at ELCE 2014
  - Presented "Overcoming Obstacles to Contributing to Linux" talk at ELC 2015
- Identify most-out-of-mainline areas
  - Analyse source tree from multiple vendors (covering multiple SOCs) for actual products, to see what the delta is in different areas
    - Initial analysis is done, and was presented at BOF in March, 2015

        - More detailed analysis is under way
- Create white paper describing obstacles
    - "Overcoming Obstacles to Mainlining White Paper (v0.9)
        - June 3, 2015

Implement solutions

- Decide on specific projects
    - Identify kernel sub-systems that are the most out-of-tree
    - Identify what the problems are
- Push patches to mainline to address problem areas found
    - Assist weary maintainers

Train and Educate participants

- Collect resource for mainlinining training
    - See Kernel Mainlining
- Convince management to provide funding and/or resources for this work
    - Produce a white paper on obstacles (see above)
    - Determine cost of out-of-tree code, and publish information for managers

Meetings

Device Mainlining BOF - October 16, 2015

[should list attendees and notes here]

Device Mainlining meeting - March 24, 2015

This meeting will be held Tuesday, March 24, at 11:00 am in the Guadalupe meeting room at the San Jose Marriott.

Attendees: [should list attendees here]

Topics:

Some of the issues I want to discuss are:

- Tim reports on his SOC research
- Where are we, really, with product-grade mainline support for SOCs?
    - Based on recent work on my part, I've come to believe we're not nearly as close to mainline support for many, many processors, as we should be. I'm in the middle of doing some research on multiple SOCs used in mobile phones, and it appears that there are all kinds of fundamental problems that make using a mainline kernel on a production phone a fleeting dream.
- What sub-systems have weakest support in mainline?
    - For each sub-system, is the poor support because of:
        - framework deficiencies
        - DT blockage
        - dependencies on out-of-tree code
        - lack of documentation
        - maintainer bandwidth, or
        - something else?
- Are there things the Linux Foundation, other industry groups, or companies can do to make interacting with the community easier?
    - Would it be helpful to produce a scorecard, for evaluating where each SOC is
        - Scorecard would have: what subsystems have been mainlined, what's outstanding, etc.

  - Can git tools be developed to automate this analysis?
  - Can Linaro and LF work together on some things?
  - keep a list of "maintainer quirks" - what things individual maintainers like/dislike, insist on, what timing they like (when to submit relative to merge window).

Calls and meetings - April and May 2015

- Tim Bird held a Birds-of-a-Feather meeting to discuss issues with this during Linux Plumbers in Dusseldorf, Germany, on October 16, 2014
  - Kevin Hillman attended, and provided some good insights into SOC mainline efforts in the past
- Tim held another meeting Special Interest Group meeting at ELC, to identify areas that need work (and could use CEWG/LF funding)
  - Tim prepared some information about the out-of-tree status of various SOC and discussed this in the meeting
- April 21, 2015 - Tim presented Obstacles to Mainlining talk at Genivi All-Members-Meeting
- April 28, 2015 - Conference call was held with Linaro (Tim Bird, Kate Stewart and Mark Brown)
  - different items were discussed, with main goal to be publication of upstream analysis tools
- Tim Bird met with Ibrahim Hadad on April 28, 2015 in San Jose
  - presented basic goals, got information about Samsung and their practices
- May 12, 2015 - Conference call with Linaro (Kate Stewart and Mark Brown)
  - had not published upstream-analysis-tools, but discussed possible names, and other topics

Obstacles (or problems to overcome)

- Version gap
- Dependency on out-of-tree code
- Inability to test (inability to generalize)
- Low-quality code
- Specialized code
- Immature upstream frameworks
- No time allocated by management
- Difficult (or missing) internal approval process
- Overloaded (or slow) maintainers
- Lack of perceived business value
- Rate of hardware churn greater than resources allocated to upstream effort
- Difficult upstream process
- Lack of English proficiency
- Fear of rejection
- Suppliers providing out-of-tree code

Categories:

- intrinsic delays in the product pipeline
- lack of corporate will
- skill deficiency
- sourcing woes??

Tools

This section has information about tools provided by this project:

Upstream Analysis Tools

Some tools that have been developed are available at: https://github.com/tbird20d/upstream-analysis-tools The purpose of these tools is multiple:

- to allow a company to analyze their own upstream status (check their own out-of-tree-ness)
- to allow a company to score the source from potential suppliers
- to try to identify areas where multiple companies have code for the same hardware or feature out-of-tree
    - this is intended to help identify where shared work makes sense
    - also, it might help identify a sub-system or framework weakness that needs addressing upstream

Some tools (also part of the release above) can be used for the following:

- to track the status of contributions from different parties to the Linux kernel over time
    - this is intended to allow an organization to monitor the progress of their mainlining activities
- to identify what their code is dependent on (commit-dep.py)
    - this is to assess the potential difficulty of mainlining a particular patch, to help in prioritizing mainlining work
    - also, this should help to identify dependencies between commits, to help in prioritizing individual patches

Resources

[put here links to training or other resources that this project develops (or should develop)]

Source code repositories

See Phones Processors and Download Sites

> From: eLinux.org

Overcoming Obstacles to Mainlining

Information about overcoming obstacles to mainlining.

Tim Bird gave a presentation about overcoming obstacles to mainlining, at ELC Europe 2014 in Düsseldorf, Germany.

Here are the slides for the talk: Media:Overcoming_Obstacles_to_Mainlining-ELCE-2014-with-notes.pdf

Please note that a few cosmetic errors were fixed, as well as 2 pages were added at the end with notes or suggestions from people I talked to at the event.

Abstract

Here is the abstract for the talk:

Many companies struggle with contributing to Open Source projects. This talk will identify key difficulties that many large companies face in making contributions, and provide tips and lessons learned for overcoming these obstacles. Some of the difficulties discussed will be: version gap, expertise problems (an example of which is the "proxy problem"), wrongly-abstracted code, process mismatch, and social and attitudinal barriers. This will not be yet another talk on CodingStyle, but a more high-level look at structural problems inside companies and the industry that prevent meaningful engagement within the open source community.

The goal of this talk is to help individual developers and companies identify and implement practices that will accelerate their participation in open source, so that they can enjoy more of the value of open source besides just the open code base.

Outline

- Intro
  - Anna Karenina Principle
    - "Happy families are all alike; every unhappy family is unhappy in its own way"
    - There are lots of ways to fail, but only a few ways to succeed

```
1. Identify obstacles
2. ???
3. Profit
(replace step 2 with "overcome obstacles"
```

- Obstacles
  - no recognition of upstream at all
    - waterfall and squirtgun metaphor
  - Version Gap
    - You can't actually help if you're working on the wrong thing
    - The kernel changes more quickly than you think
  - Expertise problems
  - Quick Hacks
    - It takes longer to write good code
    - Good code might not be better for you!
  - Bad abstractions
  - Process mismatch
    - Not using git - you're doomed
    - SubmittingPatches - death by a thousand cuts
  - Social barriers

        - misunderstanding the currency - see "And then there were none"
        - fear of failure
    - Lack of support
        - Middle managers don't get it, Execs won't prioritize it.
- Survey results
- Overcoming them
- Words from the sages
- Summary of Best Practices

Key Points

> From: eLinux.org

Qualcomm SOC Mainlining Project

Here is some information about the Qualcomm SOC mainlining project.

[this page is currently a stub]

Contents

Project List

List of items that need to be mainlined, or are "in flight" (as of June, 2014):

- regulators - Josh Cartwright?, Bjorn Andersson?
- clocks - Stephen Boyd
- RPM
- SD Card Controller - Srinivas Kandagatla
- 8084 clock controller - Georgi Djakov
- dma engine - Andy Gross
- SOC-specific
  - 8084
  - 8974
- SOC DT file? - Kumar Gala?
- USB support
  - USB host mode for HS port - Tim Bird
  - dwc3 (superspeed) USB driver
- spmi - ??

Project Table

This might be too time-consuming to maintain?

Feature or Item	Person	Notes	Link
USB - msm_otg host mode	Tim Bird	I'm currently working on the pmic_id_irq, which require spmi support. I got stuck on the DT EPROBE_DEFER support issue.	<no links yet>
SD card controller	Srinivas Kandagatla	This patch series adds Qualcomm SD Card Controller support in pl180 mmci driver. QCom SDCC is basically a pl180, but bit more customized, some of the register layouts and offsets are different to the ones mentioned in pl180 datasheet. The plan is to totally remove the standalone SDCC driver drivers/mmc/host/msm_sdcc.* and start using generic mmci driver for all Qualcomm parts, as we get chance to test on other Qcom boards.	http://thread.gmane.org/gmane.linux.ports.arm.msm/77 https://www.mail-archive.com/linux-arm-msm@vger.kernel.org/msg09059.html
DT stuff for 8084	Georgi Djakov	Adds DT nodes for the APQ8084 global clock controller and serial port.	https://www.mail-archive.com/linux-arm-msm@vger.kernel.org/msg09096.html

[add links to most recently posted patches] [add person who is working on each part]

Specific Hardware

- 8074 (Dragonboard/APQ8074)
- 8064 (Dragonboard/IFC6410, Dragonboard/SYS6440)

IP blocks on the 8974

- KRAIT - arm processor - **(DONE)**
- SAW - SPM AVS wrapper - adaptive wakup, sleep, voltage
  - AVS - adaptive voltage scaler
  - SPM - subsystem power manager - **(DONE)**
- RPM - Resource power manager - regulator voting thingy - **(in progress)** - *Bjorn, AGross*
- PM8941 - power management IC (PMIC) - see sub-parts for status
  - WLED - **(DONE)** - *Courtney*
  - PWM - pulse width modulator outputs - **(in progress)** - *Courtney*
  - LPG - (RGB LEDS) - **(in progress)** - *Courtney*
  - VIB - vibrator drivers
  - GPIO - **(DONE)**
  - MPP - **(DONE)**
  - Charger - **(in progress)** - *Courtney, Bjorn*
  - IADC - **(DONE)**
  - BMS - battery monitoring system **(not done)**
  - PON - power on thingy - **(DONE)**
  - RTC - realtime clock - **(DONE)**
  - coincell - **(in progress)** - *Tim*

    - regulators - **(in progress)** (see RPM) - *Bjorn, AGross*
    - PMIC clocks - **(not done)**
    - VADC - **(DONE?)** - *Ivan*
- PM8841 - analog regulators - **(not done)**
- MODEM - **(not done)** (but see RemoteProc)
- TLMM - top level mode manager **(DONE)**
    - PINCTRL - **(DONE)**
- QUP - configurable serial blocks
    - I2C - **(DONE)**
    - SPI - **(DONE)**
- BLSP - UART - **(DONE) (could use high speed improvements)**
- SDHCI - SD Card/emmc/EMMC controller - **(DONE - but needs regulator for HPM mode)** - *Bjorn*
- GCC - global clocks - **(DONE)**
- MMCC - multimedia clocks - **(DONE)**
- LCC - LPASS (audio) clocks - **(DONE)**
- LPASS - Audio - **(not done)** - *some 3rd party??*
- MDP - 2d display stuff - **(DONE)** - *Rob*
- SATA - **(DONE)**
- DSP - **(in progress)** - *Bjorn* - (see ??? RemoteProc??) - can send audio via slimbus, and handles sensor stuff
- SPMI - **(DONE)**
- QGIC - interrupt controller - **(DONE)**
- MPM - power manager - **(not done)**
- OCMEM - multimedia memory = **(not done)**
- USB 2.0 controller - chipidea - **(Done, but needs more)** - *Tim*
- USB 2.0 PHY - **(DONE, but needs more)** - *Tim* - needs regulator support
- HSIC PHY - **(not done)**
- USB 3.0 PHY - **(not done)**
- USB 3.0 controller - dwc3 - **(not done)**
- slimbus - peripheral bus - **(not done)**
- MI2S - multi-channel audio - **(not done)** - *Kenneth?*
- I2S - sound thingy - **(not done)**
- QCRYPTO - **(DONE)**
- PRND - random number generator - **(DONE)**
- RIVA - radio controller - **(Done, but needs major rewrite)**
- HDMI - **(DONE)**
- DSI - **(not done)**
- eDP - display port - **(not done)**
- VENUS - video codec thingy - **(not done)**
- VFE - video front end - **(not done)**
- VPE - video processing engine - **(not done)**
- CSI - camera high speed serial interface (mipi spec) - **(not done)**
- CCI - camera i2c interface (mipi spec) - **(not done)**
- TCSR - **(not done)**
    - hardware mutex - hwspinlock - **(DONE)**
        - hwspinlock DT support - **(DONE)**
    - halt control
- QFPROM - fuses - **(not done)**
- BAM - dma stuff - **(DONE - but missing utilization by individual drivers)**
- TSENSE - thermal sensing - **(in progress)**

Features

Here is a list of features (related to IP blocks) for an 8974 processor:

- charger (see charger IP block) - *Courtney*
- regulators (depends on RPM) - *Bjorn*
- remoteproc-tz - loading firmware through trustzone
- USB 2.0 gadget - **(DONE - but needs regulators)** - *Tim*
- USB 2.0 host - **(not done)**
- USB 2.0 OTG - **(in progress)** - *Tim*
- SMEM - provides an inter-processor heap - **(in progress)** - *Bjorn*
- SMD - inter-processor ring buffers - **(in-progress)** - *Bjorn*
- SMP2P - interprocessor state signaling - **(in progress)** - *Bjorn*
- SMSM - interprocessor state signaling - **(in progress)** - *Bjorn*
- Bluetooth
- FM radio
- WiFi - wcn36xx - **(in progress)** - *Bjorn*
- MHL
- NFC
- Vibrator
- Audio Codec
- IRDA
- irq_read_line - **(DONE)**
- ipcrouter - **(in progress)** - *Courtney*

Dependencies

Dependencies: '->' = "depends on"

- Regulators -> RPM -> SMD -> SMEM -> tcsr-mutex
- SMP2P -> SMEM
- SMSM -> SMEM
- SMD -> SMEM -> tcsr-mutex
- DSI -> regulators -> RPM -> SMD -> SMEM -> tcsr-mutex
- USB 2.0 controller -> charger PMIC ID -> irq_read_line
- USB 2.0 phy -> regulators -> RPM -> SMD -> SMEM -> tcsr-mutex
- USB 2.0 controller -> charger OTG switch -> charger
- charger -> irq_read_line

Resources

mailing list

- Web site: http://vger.kernel.org/vger-lists.html#linux-arm-msm
  - subscribe or unsubscribe via e-mail, by following links on the above page
- Archives
  - mail-archive.com: https://www.mail-archive.com/linux-arm-msm@vger.kernel.org/
  - gmain: http://dir.gmane.org/gmane.linux.ports.arm.msm

IRC channel

- Server: freenode (chat.freenode.net)
- Channel ##linux-msm
- Link: http://webchat.freenode.net/

You can access the IRC channel from inside a corporate firewall using the web interface.

git trees

- git trees:
  - Sony github integration ("next") tree: https://github.com/andersson/kernel/tree/next
    - Sony maintains this tree by keeping patch sets for individual technology areas or features in separate git branches. We have an internal tool called 'splash', that is used to integrate the separate branches into a single integration branch (the "next" branch). This tool is available upon request, if you want to use the same workflow as Sony.
  - Code Aurora git trees: https://www.codeaurora.org/cgit/quic/kernel
  - Linaro Qualcomm integration tree: https://git.linaro.org/landing-teams/working/qualcomm/kernel.git/shortlog/refs/heads/integration-linux-qcomlt

Other resources

- Linaro patch queue for msm (??): https://patches.linaro.org/project/linux-arm-msm/
- Linaro mainline patch queue: https://patches.linaro.org/team/linaro-landing-team-qualcomm/

Stakeholders

This is a list of parties who are (or should be) interested in the progress of this work:

- Qualcomm
- Code Aurora Forum
- Linaro
- Qualcomm Innovation Center
- Sony
- Samsung
- LG
- HTC
- Lenovo/Motorola
- Xiaomi

Table of phones, processors and download sites

For links to different download sites, see Phones Processors and Download Sites

Status

[Figure out a metric for what remains to be done]

- brainstorming ideas for metric to measure:
  - # of features out of tree (should be going down)
  - size of diff between latest msm tree and server it was based on? (should be going down)
  - linaro uses patch queue length (outstanding vs. already mainlined)

Category:

- Snapdragon

> From: eLinux.org

Session:kernel.org development and the embedded world

Contents

Session Details

Event ELC 2008 (Mountain View)

Date April 16 2008

Presenter Andrew Morton

Organization Google

Slides Media:Morton-elc-08.ppt, Media:Morton-elc-08.pdf

Video http://free-electrons.com/pub/video/2008/elc/elc2008-andrew-morton-keynote.ogg

Duration 55 minutes

Abstract

Andrew will summarize the kernel.org development and decision-making processes. Special focus will be placed upon how they impact the developers of Linux for embedded products, including the economics of using a modern kernel versus staying on a frozen older kernel version, and the economics of maintaining private patchsets versus merging work back into the public kernel. For those who choose to work with the kernel.org team, Andrew will look at how that can most effectively be done. Andrew works with Linus Torvalds and other members of the Linux development community (including open source developers and distribution vendors and industry contributors) to shepherd new features and quality improvements into the Linux kernel.

Biography

Andrew has worked at Nortel Networks' R&D labs, and Digeo Interactive, a maker of digital home entertainment products. He is currently employed by Google, working fulltime on the Linux kernel. His long experience with Linux development, and experience in the embedded realm, give Andrew a unique and valuable perspective on the issues facing embedded Liux developers.

Notes

Reports on this talk are at: http://lwn.net/Articles/278647/ and http://www.linuxfordevices.com/c/a/News/Linux-kernel-maintainer-calls-for-embedded-specialist/

Transcript

Transcribed by Tim Bird

Verified by 1 -

0:00 - 1:00:

[Introduction by Tim Bird]

1:00 - 2:00:

[Introduction continues...]

2:00 - 3:00:

[Introduction continues...]

Thanks, Tim. Yeah I actually go back in embedded a long way. I started my career as a hardware designer and I'm actually an electrical engineer.

3:00 - 4:00:

I used to do a lot of board level design, and do the operating system, the firmware, and the bringup, and device drivers, and all that sort of stuff. And it was sort-of mid-career that I started actually seriously programming these big machines.

Anybody here who was at the Monta Vista conference last year will find what I have to say today eerily familiar. And I'm sorry about that, but it bears repeating. I know my kids need to be told the same thing more than once before they'll do it.

[Slide 1 Overview] So what I was going to talk about today - I'm going to assume that the audience here are people who don't actually work on Linux for Linux' sake, per se. You in fact use Linux as part of developing some other product. It's just a tool which you use to deliver something to your customers.

What I won't be addressing today is the ongoing day-to-day interaction which contributors such as yourselves should make with the kernel.org team.

4:00 - 5:00:

That's the subject of another presentation, that I won't be covering that topic.

Certainly, we can get on with that during question and answer session. How long do we have in here, is it a full 60 minutes? ... 50 minutes. OK. There's probably about 35 or 40 minutes worth of material. I would encourage people to think of, and save up questions at the end. Generally I don't like these things to be a me-to-you. It's important that they're also you-to-me, because I am here to serve you people. You are my customers. You need to tell me how I'm going.

I'll do a bit of amateur economics about what I believe to be the economics which is unique to embedded development, and how and why that impacts the interaction that embedded developers have with the kernel.org team;

5:00 - 6:00:

Discuss what we'd like to call "patch hoarding", sitting on a pile of patches and why this can be a bad thing to do; Look at the reasons why you might be ... to merge your features into the upstream kernel rather than maintain them yourself; Look at which kernel version you might choose to use in your product development; Discuss what level of support the public kernel developers can give to you, and why they will support you in your product development; and then I'll spend a bit of time talking about how we, or more specifically I, make decisions about what things should be merged into the upstream kernel, and what the criteria are for making that decision.

[Slide 2 - The economics of Linux-for-embedded]

So looking at the economics of Linux on embedded systems, I think the key difference with embedded is that usually there is no plan to upgrade the version of the kernel running on the product.

6:00 - 7:00:

You develop the product, you build the software, the application stack, the kernel. And you bundle it all together, and mostly push it out. You may, maybe will update the kernel sometimes, but you don't expect your customer to do it. It's an embedded product.

This is radically different from the desktop and server world where upgrades are expected. In fact you don't even know what kernel your customer will be running in the first place. He might be running a RedHat Enterprise kernel, or an OpenSuse kernel, or Mandriva or anything. And also the kernel will be upgraded across the lifetime of the hardware. That's a critical difference because then you need to plan for that upgrade.

Another difference between embedded and the server/desktop world is that the software for embedded is delivered directly from the hardware manufacturer.

7:00 - 8:00: It doesn't come from some other 3rd party.

Now one exceptional case in all of this thinking is the case of the hardware manufacturer. ... that's the Intel, ARM, Atmel, Analog, people like that, who make hardware which they then sell to other customers, and those customers will then deliver embedded products using that hardware.

Those companies are in a similar position to the desktop and server hardware manufacturers, in that they are motivated to make Linux run as well as possible on their particular hardware, so that they'll pick up more customers.

So what we're seeing at present in the kernel world is a lot of contributions coming from companies such as those, who sell hardware into embedded developers, rather than actually developing embedded products themselves.

8:00 - 9:00: So that is for similar logic as you would see with the desktop and server world.

Another thing we see - and I don't really know why it is - the embedded segment just doesn't seem to have as much money. They tend to be smaller companies, with shorter time to market, and quicker turn-around times. So they can't really afford the luxury of having people who are dedicated to working on the open source software. They have to get product out the door.

The net effect of all this is that, I believe, that the embedded segment is under-represented in the development of the kernel.org kernel, compared to the overall importance of embedded in the Linux world. It's still the case that most of the people who are paid to work on Linux are really keyed towards the server products.

9:00 - 10:00:

Things have improved, mainly from the contributions from the embedded chip manufacturers. And sometimes that's indirect. For example, two of our entire embedded architectures are in fact maintained by David Howels, who's at RedHat. I expect that's on a contract basis with the companies who actually make those chips.

[Slide 4 - Effects of embedded's under-representation]

Now if my theory is correct, and embedded is in fact under-represented in kernel.org development, that will obviously have a few effects.

One is, that there will be a risk that it will tilt development to be excessively focused on the server and desktop side, and not sufficiently focused on the embedded side. And we regularly get accused of being excessively focused on servers, and not focused enough on desktop. Which, there's some truth to that, but not as much as one might think.

10:00 - 11:00:

The fact is that the people who work on kernel.org - particularly those people who you might like to call the leaders of the project - we do care a lot about Linux on the embedded system, even though we really don't have a business case for doing so. We want Linux to run as well as possible on embedded. We think it's cool. So we care about it a lot.

So a strong part of kernel development culture is any change which comes into the kernel does need to be scrutinized for its impact on small systems. That's laptops down to very small embedded devices. And if we decide a feature is too heavily oriented towards multi-processor systems, or uses too much memory, or whatever, it will need to be re-worked.

Now, we generally do this by code inspection. We don't actually do it by testing and measuring its memory footprint effects. Things can and do slip through. We could always do with some help there, reviewing and testing changes to make sure that their impact upon embedded is either zero or beneficial.

11:00 - 12:00: But, as I say, we mainly - most of the work we do looking after embedded is making sure that the work which is really geared towards server doesn't damage embedded too much. We don't actually spend much of our time overtly trying to improve embedded systems. It's just, "Let's work on server stuff, and not damage embedded too much".

And that's unfortunate.

I can't really identify anybody who sits there and works on the Linux kernel specifically to make it better for embedded systems. There're a lot of people who work down in the architecture code, and devices drivers, and all sorts of things, that are keyed towards embedded. But there is no single person I can go to who is *the* embedded maintainer.

12:00 - 13:00: Now a number of the embedded guys do work on embedded peripherally. David Howells who looks after two of the embedded architectures, for example, does a bit of work on the no-MMU port, and on various executable file formats, and things. But there's no single person sitting there on top of the Linux kernel who is the go-to person to look after embedded, to add new embedded features to the core kernel, etc.

Matt Mackall was doing it for a while, I believe under contract with CELF. But that's certainly not a full-time effort.

So, I think if we did have one person who was the embedded top level person, not oriented towards any particular architecture, that person would be pretty effective. There's a lot of work they could do to make significant benefit for the overall embedded users of the kernel.

13:00 - 14:00: [Slide 5 - How you can help kernel.org development]

Now if you are, for example, a person who is developing an embedded product for a company, that's based on Linux; If you're generally interested that Linux will run as well as possible on your product - We need your help.

And generally people think help means sitting down until 2:00 o'clock in the morning writing kernel patches and sending them to Andrew. But that's not necessarily the case. There are a lot of things that people can do which are [???] than sitting down and writing patches.

One is simply telling us what your pain points are - what the Linux kernel is not doing for you as well as you would like it to do. Often we just don't know. Some people know, for example, that boot-up time is a problem.

14:00 - 15:00: But people don't really think about it a lot while they're coding. Nobody actually sits there and tries to optimize bootup time.

We know about memory footprint - always careful about that. But we're not particularly sophisticated in addressing the needs of embedded. So it would really help if we had some people - some more senior, sophisticated people - who have experience in the embedded world, simply to tell us when we're doing things wrong. To tell us what features are important to them and what aren't. Simply so we can prioritize our work in a way that optimizes the kernel for embedded platforms.

Tell us what we're missing - I mean, I don't think I've seen anybody come up and say, "Look WindRiver, or PSOS or whatever does this, that and that, and Linux doesn't. And it's a gap."

If you think there are things that Linux should do which competitive operating systems do, well, let us know. We might not do anything about it, but at least we'd know about it and over time, people will help prioritize people's work.

15:00 - 16:00:

And, help people understand what things are beneficial to the most number of users as they work on the Linux kernel.

I find there are a lot of people who are monitoring the mailing lists, the websites, the release announcements and that sort of thing. And often I come to conferences and people tell me things I didn't know.

That's bad.

I shouldn't not know anything. If you know something which is potentially of use to the kernel team, to help guide in their development, for heaven's sake tell us. And a way to tell us is by sending an e-mail to you-know-which mailing list.

Tell us what you need.

Because there's no way that we can do it if we don't know that it's a need.

Another way in which a person can contribute to kernel.org development is to review the patches as they go past.

16:00 - 17:00:

You can help us to check that a change will not damaging to embedded in any way. You may see a feature which is close to what you need, but it needs a few tweaks to be more useful. Please tell us!

Reviewing the patches, checking them for correctness and suitability for your application is direct contribution and it's not a lot of work.

"The Squeaky gate get the oil" is an old English phrase in that the person who makes the most noise, is the one who gets the most attention. And I'm afraid that's how kernel work happens.

There's historically been a group of people who were very good at making a lot of fuss about, for example, latency on the desktop. And as a consequence, quite a lot of developers and resources have been devoted to improving latency on desktops for multimedia applications and gaming.

17:00 - 18:00:

And that wouldn't have happened if those people hadn't come on the mailing list and made us look bad. So if you want something done, come on the mailing list and make us look bad. Chances are, it will happen.

[Slide 7 - Patch hoarding]

What I see quite frequently is embedded groups and others mainly at companies working on embedded products, will tend to sit - and probably you all do this, you tend to sit on your own little pile of patches which you think are only relevant to your product, and you will maintain them out of tree. Consequently, those patches get larger and larger and uglier and uglier and older and older and they become harder and harder to merge up, and they basically never get merged into the mainline tree.

18:00 - 19:00:

It's quite a bit of work to merge a patch to into the mainline tree. I hope the main reason why people sit on their work without sending it upstream is because it's quite a lot of work to get it upstream. But I've also heard of companies who like to keep the patches to themselves because they believe they will be beneficial to their competitors.

Sorry, that's not a good reason.

This is an open source product. Yes, it will potentially be some small damage if your software becomes available to your competitors. But I think you have to see that as the price of admission to using Linux. It's an open source product. Everybody else is contributing into it. You have at the very least a moral, if not a legal obligation to contribute to it yourself.

Another problem of course with patch hoarding is that it gets increasingly expensive over time, particularly as you roll forward through kernel versions.

19:00 - 20:00: And so I would encourage people to make a big effort to get these patches merged up. And I'll be talking about that quite a bit, later on.

[Slide 8 - Why merge upstream, The pros]

Now look at the logic for what might be your reasons for merging you code up in the kernel.

We do have - it's variable, but normally we do have very strong review, integration, testing, and releasing process. So your code will be gone through with a tooth comb, and as a consequence of that, the code will be better, by the time we merge it - unless it was already very good.

A big advantage, of course, is once your code is merged with the public kernel your cost for maintaining that code falls close to zero. You no longer have to maintain a patch set. You no longer have to re-merge it and re-test it each time you have to roll forward to a new kernel version.

20:00 - 21:00:

You will find that others will add new features to it, and they will improve its performance. Other will fix bugs in it. Others will test it, and it will generally improve as time goes by. And of course when people make changes to the core kernel, rather than breaking your code, they will fix it on the fly and you will never see a difference.

A major reason for merging, particularly if it's a significant feature - another reason for merging with the mainline kernel, is that once you've merged you have won. If there are any other competing implementations of a similar feature out there, they will die. Because if you have, say, it's a particular block-based filesystem, or something like that, if you merge that, and it's a good enough block based file system, all the other ones are eventually going to wither.

21:00 - 22:00:

Now if you happen to not merge your code upstream, and somebody else merges their code, then it's your code that going to wither as well. So you're then looking at either having to maintain your alternative implementation for all time, or you'll need to migrate your existing users and applications, from your own implementation over to the one which is in the kernel, which can be quite expensive in terms of re-work and migration costs, backporting, etc.

It's not a thing which people normally think about when considering merging upstream - but being the first to merge a feature can make it easier for yourself, and harder for your competitor.

[Slide 9 - Why merge upstream? The cons]

On the other hand.

22:00 - 23:00: Just going back to that. One thing I like to do when I see somebody merging a feature is to sort of run around the industry and find out who else is interested in this feature. Who are the other stakeholders? You may find that if you and another company have a feature which do pretty similar things, but in a different way, I will try to identify that other guy. And get them to comment on your work, and find out what their take on it is. Get their input. And as a result, that feature may become more generalized, so it will address their requirements as well. That way we can possibly get them working on it also.

This has happened a number of times in the past.

Now, what might be the problems with merging your code upstream?

Obviously, it does require work. It requires a considerable body of work. Often what we see is some company will sit there in cathedral mode, and they'll develop their feature and it's all beautiful and it's ready and they'll send it off to Andrew.

23:00 - 24:00: Then the world comes tumbling down, because all sorts of things are found wrong with it. Large amounts of re-work need to be done. That's because they made the mistake of not releasing it sufficiently early - trying to get it too complete before they released it.

You should solicit feedback at the earliest possible stage so that you can get some estimate of how much work will need to be done - whether it's just completely un-viable or unacceptable, find out if somebody else has another implementation coming down the pike, etc. When looking to merge upstream you should, even if the code is not ready, get it out there for people to look at and comment on and provide you some guidance.

What can happen, I think, is people have a certain time and dollar amount set towards merging the code upstream. They budget it. They send the code, and then suddenly find that they have a couple months more work to do.

24:00 - 25:00: And they haven't budgeted for that, because it was unexpected. But, code doesn't have to be complete before we'll look at it. So get it out earlier please. It will help all parties.

Another problem with merging the code upstream can be that significant changes might be asked, or you might be asked to make significant changes to it. One thing we particularly look closely at is the user-space to kernel interfaces. We really care about interfaces. Because, the internals, you know, we can change them. We can do absolutely anything we like with the internals of the implementation. But once we've released it, one thing we cannot change is the kernel/user-space interfaces. We have to get that right.

Consequently, you may find that when you submit a significant feature into the public kernel, people will zoom in on the kernel/user interface and request that it be changed.

25:00 - 26:00: Of course this is rather nasty if you're already running that code with the particular user interface. Because you've already got all sorts of migration added to all your other problems.

Sorry. There's no real fix for that. Except for getting your code out as early as possible, so you're not too far committed to existing code before you submit ...

Another downside is, of course, a feature becomes more easily available to your competitors. I don't have a fix for that. Sorry. Except to say that's just part of the cost of doing business with Linux. It is not BSD. It's a GPL product. And, view that contribution as the price of admission.

26:00 - 27:00: [Slide 10 -

27:00 - 28:00:

28:00 - 29:00:

29:00 - 30:00:

30:00 - 31:00:

31:00 - 32:00:

32:00 - 33:00:

33:00 - 34:00:

34:00 - 35:00:

35:00 - 36:00:

36:00 - 37:00:

37:00 - 38:00:

38:00 - 39:00:

39:00 - 40:00:

40:00 - 41:00:

41:00 - 42:00:

42:00 - 43:00:

43:00 - 44:00:

44:00 - 45:00:

45:00 - 46:00:

46:00 - 47:00:

47:00 - 48:00:

48:00 - 49:00:

49:00 - 50:00:

50:00 - 51:00:

51:00 - 52:00:

52:00 - 53:00:

53:00 - 54:00:

54:00 - 55:00:

55:00 - 56:00:

56:00 - 57:00:

57:00 - 58:00:

58:00 - 59:00:

59:00 - 60:00:

Category:

- Event Session

> From: eLinux.org

Legal Issues

Contents

Legal Issues using Linux in embedded projects

The intricacies of using the GPL license have been hashed out repeatedly in many other forums.

Here are some highlights of a few specific issues that come up occasionaly:

Kernel is licensed GPL v2 only

The Linux kernel is licensed under the GNU General Public License, version 2.0 **ONLY!**

This is different from many other projects, which use the default wording in the license to allow GPL v2 or any later version.

This means it is unlikely that the kernel will switch to GPL version 3.0.

In September of 2006, a group of Linux kernel developers signed a position statement indicating that they objected to GPL version 3.0 (as then drafted). This further indicates the unlikelyhood of any change of the kernel to the GPL v3 license.

Signed-off-by lines and the DCO

When developers contribute to the kernel, they must provide a "Signed-off-by" line, indicating that they acknowledge the licensing and declare the work (to the best of their knowledge) to be either original, or derivative of something compatible with GPL v2.

See the Developer Certificate Of Origin which is contained in the kernel's Documentation/SubmittingPatches file.

Resources for legal analysis and compliance

- The Software Freedom Law Center has a compliance guide for GPL that is useful:
    - http://www.softwarefreedom.org/resources/2014/SFLC-Guide_to_GPL_Compliance_2d_ed.pdf
        - October 2014
    - Note that not everyone agrees with all legal interpretations included in this document, but overall it's a good resource.
- Copyleft and the GNU General Public License: A Comprehensive Tutorial and Guide
    - http://www.copyleft.org/guide/comprehensive-gpl-guide.html#comprehensive-gpl-guidepa1.html

EXPORT_SYMBOL_GPL

EXPORT_SYMBOL_GPL for kernel USB API

In January of 2008, Greg Kroah Hartman submitted a patch to change the core USB API to EXPORT_SYMBOL_GPL. Here is some information about that change:

- USB: mark USB drivers as being GPL only (LWN.net)
- Linux 2.6.25 without Closed Source USB Drivers (Linux Magazine)
- USB drivers going GPL-only in 2.6.25 (LinuxWorld)
- the actual git commit

Binary proprietary kernel modules

One outstanding legal question with Linux, that is of particular importance in embedded, is whether or not binary (non-GPL) kernel modules violate the GPL license of the Linux kernel.

Opinions on this topic differ.

Here is an article with some interesting information:

- Encouraging closed source modules part 1: copyright and software
- Encouraging closed source modules part 2: law and the module interface
- Encouraging closed source modules part 3: elimating the "API update tax"

Use of kernel header files in user-space

It is allowed to use kernel header files in user space, in order for user-space programs to interact with the kernel via ordinary system calls. This is allowed without the result that the user-space program becomes a derivative work of the kernel and therefore subject to GPL.

In general, use of header files do not create derivative works, although there can be exceptions. There used to be a lot of attention paid to the amount of code (e.g. number of lines) included from a header file, but no one seems to care about that these days, and this is almost never a problem. Richard Stallman has stated that use of header files for data structures, constant definitions, and enumerations (and even small inlines) does not create a derivative work. See: http://lkml.indiana.edu/hypermail/linux/kernel/0301.1/0362.html

The user-space use of the kernel header files is expected and ordinary. This explicitly encompasses non-GPL software using these files, and not being affected by the GPL. In order to calm fears about using the header files directly, and to prevent leakage of kernel internal information to user-space (where it might be abused), the mainline kernel developers added an option to the kernel build system to specifically create "sanitized" headers that are deemed safe for use by user-space programs, without incurring licensing issues.

These are the "make headers_check" and "make headers_install" targets in the kernel build system.

In general, it is legally safest to use such sanitized headers (that is, headers that have specifically been stripped of large inline macros or anything not required for user space.)

This article explains how to create sanitized kernel headers using the kernel build system. http://darmawan-salihun.blogspot.jp/2008/03/sanitizing-linux-kernel-headers-strange.html

Note that a different process was used by the developers of the Android operating system, to sanitize headers for bionic for their system. Their process was developed around the same time as the mainline header sanitization feature.

Other Links

- http://gpl-violations.org/ - The gpl-violations.org project tries to resolve GPL violations and raises public awareness about GPL compliance.

- http://www.softwarefreedom.org/ - The Software Freedom Law Center provides legal representation for open source projects and publishes information on legal issues around open source.
- http://www.linuxfoundation.org/programs/legal/compliance
  - Linux Foundation's Open Compliance Program
- http://www.binaryanalysis.org/ - A binary analysis tool for GPL compliance investigations
- http://lwn.net/Articles/386280/ - LWN.net article about the binary analysis tool (published on 2010/05/06)
- http://fossology.org/ - FOSSology is a framework to scan open source code: it currently scans for copyright and license information and can easily be extended.

Category:

- OpenSource Licensing

> From: eLinux.org

Developer Certificate Of Origin

In May 2004, the kernel development community decided to standardize on a requirement to adhere to a Developer Certificate of Origin for contributions to the Linux kernel.

The text of the DCO is located in the file Documentation/SubmittingPatches in the Linux kernel source tree.

The full text of the DCO version 1.1 (the current version as of 2011) is:

```
        Developer's Certificate of Origin 1.1

        By making a contribution to this project, I certify that:

        (a) The contribution was created in whole or in part by me and I
            have the right to submit it under the open source license
            indicated in the file; or

        (b) The contribution is based upon previous work that, to the best
            of my knowledge, is covered under an appropriate open source
            license and I have the right under that license to submit that
            work with modifications, whether created in whole or in part
            by me, under the same open source license (unless I am
            permitted to submit under a different license), as indicated
            in the file; or

        (c) The contribution was provided directly to me by some other
            person who certified (a), (b) or (c) and I have not modified
            it.

        (d) I understand and agree that this project and the contribution
            are public and that a record of the contribution (including all
            personal information I submit with it, including my sign-off) is
            maintained indefinitely and may be redistributed consistent with
            this project or the open source license(s) involved.
```

There is a kernel thread discussing the original proposal from Linus here (google groups). And here (aimsgroup).

Here is another article describing rationale for the 1.1 version: Clarifying the Developer's Certificate of Origin KernelTrap, June 14, 2005

Example

Here is an example Signed-off-by line, that indicates the submitter accepts the DCO:

```
Signed-off-by: John Doe <john.doe@hisdomain.com>
```

Older versions

The original DCO, version 1.0, read:

```
Developer's Certificate of Origin 1.0

By making a contribution to this project, I certify that:

(a) The contribution was created in whole or in part by me and I have the right
to submit it under the open source license indicated in the file; or

(b) The contribution is based upon previous work that, to the best of my knowledge,
is covered under an appropriate open source license and I have the right under that
license to submit that work with modifications, whether created in whole or in part
by me, under the same open source license (unless I am permitted to submit under a
different license), as indicated in the file; or

(c) The contribution was provided directly to me by some other person who
certified (a), (b) or (c) and I have not modified it.
```

Category:

- Categories

> From: eLinux.org

Events

Here are links to events of interest to embedded Linux developers.

Contents

Upcoming events

2015

October 2015

- Embedded Linux Conference Europe - Dublin, Ireland, October 5-7, 2015
    - http://events.linuxfoundation.org/events/embedded-linux-conference-europe

November 2015

- Regional conference in Japan - Tokyo, Japan, November 13, 2015 (Nakano Sunplaza Room 6) - Japan Technical Jamboree 54

Past events

Some information is at: Embedded linux events

Here is a list of past Embedded-related events:

2015

June 2015

- LinuxCon Japan - Tokyo, Japan, June 3-5, 2015
    - http://events.linuxfoundation.jp/events/linuxcon-japan
    - Linux Foundation archive
    - CE Workgroup Projects - LinuxCon Japan 2015
- Regional conference in Japan - Tokyo, Japan, June 19, 2015 (Nakano Sunplaza Room 6) - Japan Technical Jamboree 53

April 2015

- Regional conference in Japan - Tokyo, Japan, April 10, 2015 (Nakano Sunplaza Room 6) - Japan Technical Jamboree 52

March 2015

- Regional conference in Japan - Tokyo, Japan, March 19, 2015 (Sony corp. HQ / SAP Creative Lounge) - Japan Mini Technical Jamboree
- Embedded Linux Conference - San Jose, California, March 23-25, 2015
    - http://events.linuxfoundation.org/events/embedded-linux-conference
    - Linux Foundation archive
    - Presentations

- Videos of Presentations: YouTube

2014

December 2014

- Regional conference in Japan - Tokyo, Japan, December 19, 2014 (Nakano Sunplaza Room 6) - Japan Technical Jamboree 51

October 2014

- Embedded Linux Conference Europe - Germany, Dusseldorf, October 13 - 15, 2014
  - http://events.linuxfoundation.org/events/embedded-linux-conference-europe
  - Linux Foundation archive
  - ELC Europe 2014 Presentations
- Regional conference in Japan - Tokyo, Japan, October 24, 2014 (Nakano Sunplaza Room 6) - Japan Technical Jamboree 50

September 2014

- Fossetcon 2014 - Free and Open Source Software Expo and Technology Conference, Orlando, FL, September 11 - 13, 2014
  - http://fossetcon.org

July 2014

- Regional conference in Japan - Tokyo, Japan, July 25, 2014 (Nakano Sunplaza Room 8) - Japan Technical Jamboree 49

May 2014

- LinuxCon Japan 2014, Tokyo, May 20-22, 2014
  - http://events.linuxfoundation.org/events/linuxcon-japan
  - Linux Foundation archive
- Regional conference in Japan - Tokyo, Japan, May 23, 2014 (Nakano Sunplaza Room 6) - Japan Technical Jamboree 48

April 2014

- Embedded Linux Conference 2014, San Jose, CA, April 29-May 1, 2014
  - http://events.linuxfoundation.org/events/embedded-linux-conference
  - Linux Foundation archive
  - Presentations at: ELC 2014 Presentations
  - Videos of Presentations: ELC 2014 Presentations

2013

December 2013

- Regional conference in Japan - Tokyo, Japan, December 11, 2013 (Nakano Sunplaza Room 2) - Japan Technical Jamboree 47

October 2013

- Embedded Linux Conference Europe 2013, Edinburgh, UK, October 24 - 25, 2013
  - http://events.linuxfoundation.org/events/embedded-linux-conference-europe
  - Linux Foundation archive
  - ELC Europe 2013 Presentations

September 2013

- Regional conference in Japan - Tokyo, Japan, September 13, 2013 (Nakano Sunplaza Room 2) - Japan Technical Jamboree 46

August 2013

July 2013

June 2013

- Regional conference in Japan - Tokyo, Japan, June 7, 2013 (Nakano Sunplaza Room 2) - Japan Technical Jamboree 45

May 2013

- LinuxCon Japan 2013 - May 29-31 Tokyo, Chinzan-so Hotel and Conference Center
  - LinuxCon Japan 2013
  - Linux Foundation archive

March 2013

- Regional conference in Japan - Tokyo, Japan, March 8, 2013 (Nakano Sunplaza Room 2) - Japan Technical Jamboree 44

February 2013

- FOSDEM - Feb 2-3

Brussels, Belgium

- Buildroot Developer Days - Feb 4-5

Brussels, Belgium

- Android Builders Summit

San Francisco, CA, Feb 18-19, 2013

-
  - Linux Foundation archive
- Embedded Linux Conference] - San Francisco, CA, Feb. 20-22, 2013
  - http://events.linuxfoundation.org/events/embedded-linux-conference
  - Linux Foundation archive
  - Presentations at: ELC 2013 Presentations
  - Videos of Presentations: Videos by Free Electrons

2012

December 2012

- Regional conference in Japan - Tokyo, Japan, December 7, 2012 (Nakano Sunplaza Room 2) - Japan Technical Jamboree 43

November 2012

- Embedded Linux Conference Europe - Barcelona, Spain, November 5-7, 2012
    - http://events.linuxfoundation.org/events/embedded-linux-conference-europe
    - Presentations at: ELCE Europe 2012 Presentations
    - Videos of Presentations: ELCE 2012 Videos
- LinuxCon Europe - Barcelona, Spain, November 5-7, 2012
    - http://events.linuxfoundation.org/events/linuxcon-europe

September 2012

- Automotive Linux Summit in Warwickshire, England September 19th and 20th
- Regional conference in Japan - Tokyo, Japan, **September 20 (date changed)**, 2012 (Nakano Sunplaza Room 2) - Japan Technical Jamboree 42

August 2012

- LinuxCon North America - San Diego, California, August 26-28, 2012
    - http://events.linuxfoundation.org/events/linuxcon

July 2012

- O'Reilly Open Source Convention (OSCON) at Portland, Oregon Convention Center July 16-20, 2012. The Call for Papers is scheduled to be posted in November 2011. Please add your suggestions for a proposed OSCON 2012 Embedded Linux track (or a satellite meeting)?
- RMLL - Geneva, Switzerland, July 7-12,
    1. Call for Paper is open : http://2012.rmll.info/en/participate/call-for-papers , the deadline for submission is 31 March 2012.
-
    1. Embedded-Linux Week in Wuerzburg, Germany, July 9-13, 2012. http://www.linux4embedded.de/de

June 2012

- Regional workshop in Japan - Osaka, Japan, June 15, 2012 (Osaka, ATC Hall Room B5) - LTSI workshop in Osaka
- Regional conference in Japan - Tokyo, Japan, **June 21 (date changed)**, 2012 (Nakano Sunplaza Room 1) - Japan Technical Jamboree 41
- LinuxCon Japan - Yokohama, Japan, June 6-8, 2012
    - http://events.linuxfoundation.org/events/linuxcon-japan

April 2012

- Collaboration Summit - San Francisco, California, April 3-5, 2012
    - http://events.linuxfoundation.org/events/collaboration-summit

March 2012

- Regional conference in Japan - Tokyo, Japan, March 23, 2012 - Japan Technical Jamboree 40

February 2012

- Fosdem - Brussels, Belgium, February 4-5, 2012 - http://fosdem.org/2012/embedded-devroom-cfp
- Linaro Developer Summit - Redwood Shores, California, February 6-10, 2012
- Android Builders Summit - Redwood Shores, California, February 13-14, 2012 - http://events.linuxfoundation.org/events/android-builders-summit
- Embedded Linux Conference - Redwood Shores, California, February 15-17, 2012
  - http://events.linuxfoundation.org/events/embedded-linux-conference
  - Presentations

2011

- Regional conference in Japan - Tokyo, Japan, December 9, 2011 - Japan Technical Jamboree 39
- LinuxCon Brazil - São Paulo, Brazil, November 17-18, 2011
- Kernel Summit 2011
  - Prague, Czech Republic, October 23-25, 2011
  - ARM Subarchitecture Maintainership Workshop - Events/Kernel Summit 2011 ARM Subarch Maintainership Workshop
- Embedded Linux Conference Europe - Prague, Czech Republic, October 26-28, 2011
  - Linux Foundation archive
  - Presentations at: ELCE_2011_Presentations
  - ELCE 2011 Technical Showcase
  - Videos recorded and encoded by Free Electrons
  - Long Term Support Kernel Meeting 2011
    - this is a private meeting for discussion about the LTSI project by the CEWG
- LinuxCon Europe - Prague, Czech Republic, October 26-28, 2011
  - http://events.linuxfoundation.org/events/linuxcon-europe
- Regional conference in Japan - Tokyo, Japan, September 30, 2011 - Japan Technical Jamboree 38
- LinuxCon North America 2011 - Vancouver, Canada, August 17-19, 2011 -
- LinuxCon Japan - Yokohama, Japan, June 1-3, 2011
  - http://events.linuxfoundation.org/events/linuxcon-japan
- Regional conference in Japan - Tokyo, Japan, May 20, 2011 - Japan Technical Jamboree 37
- Android Builders Summit (ABS) - San Francisco, April 13-14, 2011
  - Presentation slides for ABS 2011
  - Videos, recorded and encoded by Free Electrons
- Embedded Linux Conference] (ELC) - San Francisco, April 11-13, 2011

  - Linux Foundation archive
  - Presentations at: ELC 2011 Presentations
  - Videos, recorded and encoded by Free Electrons
- linux.org.au Conference -- Brisbane, Queensland, Australia. 16th-22nd of January 2010.

- Regional conference in Japan - Tokyo, Japan, March 18, 2011 - Japan Technical Jamboree 36

  - **Due to unstable electrical and public transportation situation in Greater Tokyo area, this event was canceled.**
- Southern California Linux Expo (SCALE), February 18-20, 2011.

- RMLL - Strasbourg, France, July 9-14, 2011
  - Features an Embedded Systems and Open Hardware topic.

2010

- Regional conference in Japan - Tokyo, Japan, December 10, 2010 - Japan Technical Jamboree 35
- Linux Plumbers Conference 2010 - Cambridge, MA on November 3-5, 2010.
- Embedded Linux Conference Europe - Cambridge, UK, October 27-28, 2010
  - ELC-Europe

    - Presentations at: ELC Europe 2010 Presentations
- LinuxCon Japan 2010
    - Tokyo, Japan, September 27-29, 2010
    - Embedded Developer BoF 2010
        - (same place as LinuxCon Japan), September 28, 2010
- Regional conference in Japan - Tokyo, Japan, September 3, 2010 - Japan Technical Jamboree 34
- LinuxCon North America 2010 - Boston, Massachusetts, USA, August 10-12, 2010
- Embedded Systems Conference India - Bangalore, July 21-23, 2010
    - Home page - http://www.esc-india.com
    - Get Connected - http://www.esc-india.com/form.html
    - Venue - NIMHANS Convention center, Bangalore
- 12th Annual Linux Symposium - Ottawa, Canada, July 13th-16th, 2010
- RMLL - Bordeaux, France, July 6-11, 2010
    - Features an Embedded Systems and Open Hardware topic.
- Regional conference in Japan - Tokyo, Japan, June 4, 2010 - Japan Technical Jamboree 33
- Embedded Linux Conference 2010 - San Francisco, California, USA, April 12-14, 2010
    - Home page - http://www.embeddedlinuxconference.com/elc_2010/index.html
    - Call for Presentations - ELC 2010 Call for Presentations
    - Presentations at: ELC 2010 Presentations
    - Videos of Presentations
- Regional conference in Japan - Tokyo, Japan, March 5, 2010 - Japan Technical Jamboree 32
- Embedded World 2010 - Nürnberg, Germany, March 2-4, 2010 - Embedded World 2010
- GStreamer_2010_Presentations
- Embedded_Linux_Summit_2010
- ELCE_2010_Technical_Showcase

2009

- Regional conference in Japan - Tokyo, Japan, December 18, 2009 - Japan Technical Jamboree 31
- Korean Technical Conference - Seoul, Korea, November 6, 2009 - Korea Technical Jamboree 5
- Japan Linux Symposium - Tokyo, Japan, October 21-23, 2009
    - Home page - http://events.linuxfoundation.org/events/japan-linux-symposium
        - Japan Linux Symposium 2009 for Embedded System Developers (A special note for embedded system developers).
- CE Linux Forum Project BoF and Plenary meeting - Tokyo, Japan, October 22, 2009
    - Colocated with Japan Linux Symposium
    - CELF_BOF_and_Plenary_2009
- Embedded Linux Conference Europe 2009 - Grenoble, France, October 15-16, 2009
    - Home page - http://www.embeddedlinuxconference.com/elc_europe09/index.html
    - Videos
    - All presentations at ELC_Europe_2009_Presentations
- Regional conference in Japan - Tokyo, Japan, October 2, 2009 - Japan Technical Jamboree 30
- Regional conference in Japan - Tokyo, Japan, July 17, 2009 - Japan Technical Jamboree 29
- Linux Symposium 2009 - Montreal, Canada, July 15-18, 2009
    - Home page - http://www.linuxsymposium.org/2009/
    - Measuring Function Duration With Ftrace Presentation by Tim Bird
        - Ftrace Function Graph ARM
            - presentation, paper, patches and ftd tool from talk
    - Embedded Linux Status Presentation by Tim Bird
- RMLL, Nantes, France, July 7 to 11
    - Featured an *Embedded systems and open hardware* track (see the call for papers)
        - Videos and papers
- Regional conference in Japan - Osaka, Japan, June 12, 2009 - Japan Technical Jamboree 28

- FreedomHEC Taipei - Taipei, Taiwan, June 10 ~ 11, 2009
- Regional conference in Japan - Tokyo, Japan, May 22, 2009 - Japan Technical Jamboree 27
- Embedded Linux Conference 2009
  - San Francisco, California, April 6-8, 2009
  - Home page - http://www.embeddedlinuxconference.com/elc_2009/
  - Presentations - ELC_2009_Presentations
- Regional conference in Japan - Tokyo, Japan, March 26, 2009 - Japan Technical Jamboree 26
- Free Open "Embedded Linux" Training for Students in India - February 20 and 21, 2009
  - Home Page - beagleboard.org Trainings in India
- Regional conference in Japan - Osaka, Japan, January 30, 2009 - Japan Technical Jamboree 25

2008

- Regional conference in Japan - Tokyo, Japan, March 26, 2009 - Japan Technical Jamboree 24
- Regional conference in Korea - Seoul, Korea, November 21, 2008 - Korea Technical Jamboree 4
- FreedomHEC Taipei 2008, Taipei, November 20-21, 2008
- Embedded Linux Conference Europe - Ede, The Netherlands, November 6-7, 2008
  - Home page - http://www.embeddedlinuxconference.com/elc_europe08/index.html
  - ELC Europe 2008 Presentations page.
  - Videos page.
- Open source DLNA Summit in Ede, The Netherlands, November 8, 2008
- Regional conference in Japan - Tokyo, Japan, October 30, 2008 - Japan Technical Jamboree 23
- BarCamp Bangalore - Bangalore, India, 13th September 2008
  - Home Page - http://barcampbangalore.org/wiki/BCB7_Demos
- IEEE SP Connect 2 - NITK Suratkal, India, August 30 2008
  - Home Page - http://www.nitkieee.com/site/sp-connect2/schedule
- Regional conference in Japan - Tokyo, Japan, August 29, 2008 - Japan Technical Jamboree 22
- Ottawa Linux Symposium 2008 - Ottawa, Canada, July 23-26
  - Home page: http://www.linuxsymposium.org/2008/
  - Embedded Linux BOF Presentation by Tim Bird
  - OLS 2008 CELF Embedded Developer BOF
- Regional conference in Korea - Seoul, Korea, July 11, 2008 - Korea Technical Jamboree 3
- Regional conference in Japan - Tokyo, Japan, July 4, 2008 - Japan Technical Jamboree 21
- Regional conference in Japan - Tokyo, Japan, April 25, 2008 - Japan Technical Jamboree 20
- Embedded Linux Conference, 2008 - Mountain View, California, USA, April 15-17, 2008
  - ELC 2008 Presentations
  - Videos page.
- Regional conference in Japan - Tokyo, Japan, February 21, 2008 - Japan Technical Jamboree 19
- Regional conference in Korea - Seoul, Japan, February 22, 2008 - Korean Technical Jamboree 2
- RMLL 1 to 5 july ,Mont-de-Marsan France
  - Videos

2007

- Regional conference in Japan - Tokyo, Japan, December 21, 2007 - Japan Technical Jamboree 18
- Embedded Linux Conference - Europe 2007 - Linz, Austria, November 2-4, 2007
  - Presentations have been collected and posted at ELC Europe 2007 Presentations
- Regional conference in Japan - Tokyo, Japan, October 25, 2007 - Japan Technical Jamboree 17
- Regional conference in Japan - Tokyo, Japan, August 31, 2007 - Japan Technical Jamboree 16
- Ottawa Linux Symposium 2007 - Ottawa, Canada, June 27-30, 2007
  - Ottawa Linux Symposium 2007
  - OLS 2007 Embedded Linux BOF
    - June 27

  - OLS 2007 Embedded Linux Wiki BOF
    - June 27
  - OLS 2007 CELF BOF - June 29, 7-9pm, Westin Hotel
- Regional conference in Japan - Tokyo, Japan, June 29, 2007 - Japan Technical Jamboree 15
- Embedded System Expo & Conference in Japan - Tokyo, Japan, May 16-18
- Regional conference in Japan - Tokyo, Japan, April 27, 2007 Japan Technical Jamboree 14
- Embedded Linux Conference, 2007 - San Jose, California, April 17-19, 2007 ELC 2007
  - ELC_2007_Call_For_Presentations
  - Presentations are available at ELC_2007_Presentations
- Regional conference in Japan - Tokyo, Japan, February 22, 2007 - Japan Technical Jamboree 13

2006

- Regional conference in Japan - Tokyo, Japan, December 8, 2006 - Japan Technical Jamboree 12
  - (Page copied locally at: Japan Technical Jamboree 12
- Embedded Technology 2006 - Yokohama, Japan, November 15..17,
- Regional conference in Japan - Tokyo, Japan, October 27, 2006 - Japan Technical Jamboree 11
- Regional conference in Japan - Tokyo, Japan, August 25, 2006 - Japan Technical Jamboree 10
- GNOME Embedded and Mobile (GMAE) forum first meetting at Guadec 2006 in Villanova, Spain. Have a look at the report: Gmae Report for CELF 2006
- Ottawa Linux Symposium 2006 - Ottawa, Canada, July 19-22, 2006
  - We had a great time in Ottawa! See the following pages for more information:
  - Ottawa Linux Symposium 2006
  - OLS 2006 CELF Demos
- Regional conference in Japan - Tokyo, Japan, July 13, 2006 - Japan Technical Jamboree 9
- Embedded System Expo & Conference in Japan - Tokyo, Japan, June 28-30, 2006
  - Information for visitors to CELF booth: Japan ESEC 2006
- Regional conference in Japan - Tokyo, Japan, May 26, 2006 - Japan Technical Jamboree 8
- CELF Embedded Linux Conference - San Jose, California, April 11-12, 2006 - elc2006
  - Presentations from the conference are being collected at: ELC 2006 Presentations
  - Biographies of ELC 2006 presenters at: ELC 2006 Biographies
  - Instructions for visitors from Japan: Japan Jamboree To WELC 2006
- Regional conference in Japan - Tokyo, Japan, March 24, 2006 Japan Technical Jamboree 7
- Regional conference in Japan - Tokyo, Japan, January 20, 2006 Japan Technical Jamboree 6
- [RMLL http://2006.rmll.info/] : 4 to 8 july 2006, Vandoeuvre-les-Nancy,France
  - Videos

2005

- Regional conference in Japan - Tokyo, Japan, November 25, 2005 Japan Technical Jamboree 5
- Regional conference in Japan - Tokyo, Japan, September 30, 2005 Japan Technical Jamboree 4
- Ottawa Linux Symposium - Ottawa, Canada, July 20-23 - OLS
  - CELF was a sponsor again this year for the Ottawa Linux Symposium. We were permitted to show some demos at the event. CELF held a Birds-of-a-Feather (BOF) session on embedded Linux. Also, CELF handed out some prizes at the final address.
- Regional conference in Japan - Tokyo, Japan, July 15, 2005 Japan Technical Jamboree 3
- International technical conference - Yokohama, Japan, June 2005 International Technical Jamboree
- Regional conference in Korea - Seoul, Korea, May 14, 2005 - CELF Korea Tech Conference
- Linux Conf Australia - Canberra, Australia, April 18, 2005
  - A few individuals attended, and Tim put up a poster of bootup-time results
- CELF Worldwide Technical Conference - San Jose, January 25, 26, 2005
  - Conference program: CELF Technical Conference 2005
  - (local copy here: Technical Conference 2005)

  - Presentations from various sessions: Tech Conference 2005 Docs
- RMLL : July,Dijon, France
  - Videos and papers

2004

- Regional conference in Japan - December, 2004 - Japan Technical Jamboree 2
- Regional conference in Japan - October, 2004 - Japan Technical Jamboree
- Geek_Cruises
- Ottawa Linux Symposium, July 2004 - Tim gave a talk, and CELF held a BOF and gave out prizes
  - see OLS2004

Links to Papers from other events

- Fosdem 2005 embedded kernel papers - FOSDEM
- Ottawa Linux Symposium proceedings - OLS
  - OLS 2007 - broken out paper links - https://ols2006.108.redhat.com/2007/Reprints/
- OLS proceedings collected on kernel.org - http://kernel.org/doc/ols/
- Linux from naught to 60 in 5 seconds - Auke Kok & Arjan van de Ven

Presentations at Kernel Summits

- See Kernel Summit 2009

Event Planning Pages

- Event Planning Pages

ELC Conference Presentations

- ELC_Presentations

Category:

- Events

From: eLinux.org

CELF BOF and Plenary 2009

This page describes the CELF Bird's-of-a-Feather meeting and Plenary meeting.

This meeting will consist of a reception, lightning discussions, and general mingling, followed by a formal CELF Plenary meeting.

Members of the public are welcome to attend the BOF and reception, while members of the forum (employees of CELF members companies) may attend the Plenary meeting.

Contents

Meeting Details

- Date: October 22nd
- Time: 19:00 - 21:00
- Location: Ochanomizu Hotel Juraku
- Flier: (pdf file) Media:projectBoFflyer.pdf



- The Bof meeting scene

- Date: October 22nd
- Time: 19:00 - 21:00
- Location: Ochanomizu Hotel Juraku
- Flier: (pdf file) Media:projectBoFflyer.pdf



- The Bof meeting scene



- Thomas Gleixner to present about the latest situation of real-time preempt patches to the mainline.



Agenda

Birds-of-a-Feather

The BOF and reception starts at 19:00 (7:00 pm) and runs to 20:30. There will be light snacks and drinks served, and an opportunity to present lightning talks.

Directions to Meeting-place Hotel

A map of access to the hotel is available at:

- http://www.juraku.com/ocha/english/access.htm
- File:Print HotelJyurakuMap LRs.pdf (From main venue of Japan Linux Symposium)

From Narita:

- Take JR Line for Tokyo (Narita Express), then...
- transfer to Yamanote or Keihin-Tohoku line north bound to Akihabara Station, *OR*
- transfer to Chuo line to Ochanomizu station

both are just few minutes ride.

If it is commuter time (7am to 10am / 5pm to 8pm) I highly recommend you to take taxi from Tokyo station which may cost around 1000 Yen.

- Take Keisei Line for Keisei-Ueno (Skyliner), then...
- Transfer at Nippori station to south bound of Yamanote or Keihin-Tohoku line to Akihabara station. It is just few minutes ride, *OR*
- Take taxi at Keisei-Ueno Station, the final stop. It may cost about 1000 Yen.

From Haneda:

- Take monorail to Hamamatsucho station and transfer to Yamanote or Keihintohoku line north bound to Akihabara Station. It is about 10 minutes ride.

More pictures from the event



- Ueda-san talking about Linus



- Kumagai-san, of Sharp



- Another picture of Thomas Gleixner

Categories:

- Events
- 2009

CELF Korea Tech Conference

The First CELF Korea Tech Conference were held.

It was a very successful event. The number of participants were more than expected, feedbacks were also very positive.

- Date: May 14, 2005
- Venue: Bit Campus(Seoul, Korea)
- PR support: KLDP(http://kldp.org), KELP(http://kelp.or.kr), KESIC(http://kesic.or.kr)
- Media Support: Embedded World (a Korean monthly magazine for Embedded Technology)
- Number of Participants: about 250 (overbooked)
- Picture Gallery(unofficial): http://gallery.kldp.org/upload
- Moving Picture(unofficial): http://people.kldp.org/~akpil/CELF2005/

Presentation Material	Session Description	Person	Company
Media:TargetBuilder-ETRI.pdf	Target Builder Introduction	Pyong-Jae Jung	ETRI
Media:RealTime-ETRI.pdf	Realtime in Linux	Yong-Kwan Lim	ETRI
Media:AVNetworking-LG.pdf	Home AV Networking & Linux	Geon-Ho Lee	LG
Media:ParallelPort-LG.pdf	Controlling Device using Parallel Port	Hyo-Jun Lim	LG
Media:ARMuClinux-Samsung.pdf	Merging Linux/uClinux 2.6 & Benchmark	Hyok S. Choi	Samsung
Media:PowerMgmt-Samsung.pdf	Low Power Linux Platform Development	Min-Sung Jang	Samsung
Media:JoinOpenSource-Sony.pdf	How to Join Open Source	Tim Bird	Sony

Category:

- Events

> From: eLinux.org

CE Workgroup Projects - LinuxCon Japan 2015

The following projects are being presented by the CE Workgroup at LinuxCon Japan 2015

Please come by our booth to talk to CE Workgroup representatives about these projects. Our booth is located just outside the keynote hall.

Contents

LTSI Test Project

- An Open-source test framework for the LTSI kernel
- See the presentation "LTSI: Latest Status and Kernel Testing" from ELC 2015
- See project details at: http://ltsi.linuxfoundation.org/ltsi-test-project
- Code for the test framework can be downloaded from: https://bitbucket.org/cogentembedded/jta-public/

Shared Embedded Linux Distribution

- This is a project to use Debian packages with the Yocto Project
  - Goal is to share the work of maintaining long-term support for an embedded distribution, by leveraging the work of the Debian project
  - The elinux page for this project is: Shared Embedded Linux Distribution
- See "Poky meets Debian: Understanding How to Make an Embedded Linux by Using an Existing Distribution's Source Code" talk at ELC 2015 by Yoshitake Kobayashi

Device Mainlining Project

- This project aims to reduce the amount of out-of-tree code for Consumer Electronics projects (particularly mobile devices)
  - Research shows that, on average, a mobile product based on Linux and Android has about 2 million lines of code out-of-mainline
  - The main web page for this project is: CE Workgroup Device Mainlining Project
  - Key activities include:
    - Promotion of best practices for corporate guidance (e.g. the white paper)
    - Collection and organization of links to mainlining tutorials and training materials
    - Development of tools and information to assist companies with mainlining-related tasks
    - Mainlining status analysis tools (see https://github.com/tbird20d/upstream-analysis-tools)
    - Quantification of costs associated with out-of-tree code
    - Providing assistance for upstream maintainers
- The new "Overcoming Obstacles to Mainlining" white paper is published at LinuxCon Japan 2015 (June 3, 2015)
  - Based on the presentation "Overcoming Obstacles to Contributing to Linux" presented at ELC 2015 by Tim Bird

Linux in Infrastructure

- This project aims to address requiremens for using Linux in civil and societal infrastructure projects
- Presentation material is available at the following.
  - "Applying Linux to the Civil Infrastructure"

eLinux wiki

- The CE Workgroup continues to support this wiki, which has lots of information valuable to embedded Linux developers
  - This includes technical presentations from the last 10 years of Embedded Linux Conference events, as well as wiki pages for lots of technical areas and CEWG's projects

From: eLinux.org

DLNA Summit 2008

Contents

Introduction

The CE Linux Forum is hosting the second Open Source DLNA Summit, in Ede Netherlands, November 8 2008.

This summit is co-located at the same venue as Embedded Linux Conference Europe.

```
(Note that registration in Embedded Linux Conference is not required for attending the DLNA Summit)
```

Slides from the Event

CELF Coherence Talk

Coherence-Jabber/XMPP

GUPnP Presentation

Purpose

The primary purpose of this conference is to facilitate the meeting and sharing of information among several parties interested in open source implementations of the DLNA specifications. These includes open source developers, product developers who might use these implementations, representatives from the DLNA organization or from related companies (certification and testing labs), and from members of the CE Linux Forum who are interested in this work.

We shall also be conducting a DLNA plug-fest during this summit. Particpants interested in the plugfest are invited to bring their DLNA devices to test for interoperability with other devices.

This page has information about the summit. It will be updated as more information becomes available.

The CE Linux Forum is interested in promoting and supporting work in this area, for eventual use in Linux-based CE products. It is hoped that by collaborating we can avoid duplication of effort, strengthen interoperability between implementations, and help accelerate the development of robust and feature-full solutions. During this summit we also want to gauge the interest level for holding such events in the future and discuss ideas for making them more productive.

Overview

- Location: Ede, Netherlands

- Date: November 8 (Saturday), 2008
- Times: 10 am to 5 pm
- Venue: This summit is co-located at the same venue as Embedded Linux Conference Europe.
- Price: Free
- Attendees: Attendance is by invitation. If you are interested in attending, please contact Rahul Saxena: rahul dot saxena at intel.com or Tim Bird: tim dot bird at am.sony.com

Agenda

List of *possible* presentation and discussion topics:

- Specification Roadmap
- Status of DLNA OSS Projects
- Industry input on DLNA stack feature priorities and problem areas
- CELF sponsorship and support of OSS projects, contract work etc
- Open sourcing of DLNA stacks from CE companies
- Ideas on collaboration: Code sharing, partitioning problem space etc
- Certification Processes and Tools
- Hardware availability: Development and Test hardware
- DLNA Plug Fest
- Share compliance experiences e.g. PS3 war stories
- Future Summit discussion: Need for such a summit, agenda Ideas, mail lists, new invitees etc

Short (up to 30 minutes) presentations on above or similar topics are welcome and encouraged. Though Pre-submission of presentations is not required, please inform Rahul Saxena (rahul dot saxena at intel.com) if you plan to present.

Attendees

Attendees: Attendance is by invitation. If you are interested in attending, please contact Rahul Saxena: rahul dot saxena at intel dot com

Current List of Attendees

Attendee	Company	Email
Tim Bird	Sony, Chair of CE Linux Forum Architecture Group	Tim dot Bird (at) am dot Sony dot com
Rahul Saxena	Intel	Rahul dot Saxena (at) Intel dot com
Frank Scholz	Beebits	fs (at) beebits dot net
Olivier Carmona	AWOX, Chair of the DLNA Certification and Test Committee	ocarmona (at) awox dot com
Philippe Normad	Fluendo	philippe (at) fluendo dot com
Gustavo Sverzut Barbieri	Profusion Embedded Systems	barbieri (at) profusion dot mobi
Christian Schaller	Collabora	christian dot schaller (at) collabora dot co dot uk
Zeeshan Ali	Nokia	zeeshan dot ali (at) nokia dot com
Jussi Kukkonen	Intel	jku (at) linux dot intel dot com
Ruud Derwig	NXP	ruud dot derwig (at) nxp dot com
Armijn Hemel	Loohuis Consulting	armijn (at) uulug dot nl
Young Sun Park	LG Electronics	yspark (at) lge dot com

Venue/Hotel Information

This summit is co-located at the same venue as Embedded Linux Conference Europe (ELCE). The summit will be located in:

```
Hotel en Congrescentrum De Reehorst
Bennekomseweg 24
6717 LM EDE
0318-75030
http://www.reehorst.nl
```

Link to more resources

More information about open source DLNA projects is available on the eLinux wiki at:

DLNA_Open_Source_Projects

DLNA_Hardware_Sharing_List

Link to 2007 DLNA Summit website:

http://tree.celinuxforum.org/CelfPubWiki/DLNASummit

Categories:

- Events
- 2008

> From: eLinux.org

ELC 2006 Biographies

Biographies for ELC 2006 go here:

Contents

Andre Kruetzfeldt

Armin Gerritsen

Armin Gerritsen is a young and enthusiastic embedded software developer. He started as a physicist, but entered Philips shortly after receiving his master degree on the area of computational physics, where he was hired to do modelling for future generations of Philips Semiconductor platforms. Shortly after he was sucked into the embedded Linux world and failed to escape since then. His job is to support the implementation of new designs by writing software necessary for (dis)proving new concepts, prototypes and demonstrator platforms. The vast majority of this software is based on Linux.

Chang-Sik Cho

Denis Olivier Krop

Ed Plowman

Ed Plowman has more than 10 years of experience in media and graphics processing. Ed is currently employed as 3D Graphics Products Manager with ARM Ltd, guiding ARM's 3D graphics processing strategy. He also represents ARM in the OpenGL ES working group and occupies the ARM seat on the Khornos board of promoters. Ed started his career as one of the founding members of Argonaut technologies (an off shoot of Argonaut Software which later became ARC cores), working on CPU based 3D graphics technology which later provided the foundation for the ARC processor core.

Greg Kroah-Hartman

Greg Kroah-Hartman is the current Linux kernel maintainer for more driver subsystems than he wants to admit, along with the driver core, sysfs, kobject, kref, and debugfs code. He is also the maintainer of the linux-hotplug and udev projects. On top of that he maintains the Gentoo Linux packages for these programs, and helps with the kernel package, which put him in touch with actual users every day, thereby ensuring his email inbox is never empty. He is also the co-author of the book, "Linux Device Drivers, 3rd Edition" and a contributing editor for Linux Journal.

Greg Ungerer

Greg specializes in embedded systems development, having spent more than 15 years developing embedded software, and more than 10 years developing with Linux.

Greg is currently one of the core maintainers of the uClinux project. He first started hacking on it 7 years ago, and did the port to the Motorola Cold Fire CPU family. Over the last couple of years Greg has been pushing the uClinux support into Linus's mainline 2.6 series Linux kernels. Greg has been involved in the design and release of many commercial products based on uClinux software over the last few years.

Hyok S. Choi

Hyok S. Choi is the maintainer of uClinux/ARM and a system architect for electronic gadgets in Samsung. The project URL is http://opensrc.sec.samsung.com/

Jared Hulbert

Jared Hulbert has been working with embedded Linux since 2000 when he picked up a ucSimm. Hes worked with XIP on Linux since 2001 helping port it to various machines. He now works for the CTO of the Intel Flash Memory Group managing the Linux Team trying to improve Intel NOR and NAND flash memory support for Linux. Jared has helped Intel customers optimize and port Linux on several phones currently on the market. He has also created many systems as demos for customers or internal uses such as what was probably the worlds only XIP Linux iPaq.

John Goodacre

Program Manager, Multiprocessing, ARM

John joined ARM in February 2002 taking responsibility for their platform architecture roadmap. More recently he has been responsible for the market development of multiprocessing technology and the release of ARM11 MPCore the first integrated SMP core.

Prior to working at ARM, he worked for Microsoft for 5 years, firstly as Group Program Manager in the Exchange Server group and latterly as the manager for its Wireless Telephony group responsible for product, the definitions and strategy of mobile devices.

Graduating from the University of York with a BSc in Computer Science, John has over 20 years experience in the engineering industry.

Jordan Crouse

Jung-Hyun Yoo

Kevin D. Kissell

Kevin D. Kissell is Principal Architect at MIPS Technologies, and has been a part of the MIPS architecture development team since its spin-out from Silicon Graphics in 1997. He was first involved in the architecture and design of RISC microprocessors when he joined the original Fairchild Clipper design team in 1983. In between, Kevin has been variously responsible for processor, systems and software architecture, for decoupled access/execute supercomputers at ACRI, massively parallel distributed memory computers at nCUBE, and large-scale shared-memory supercomputers at Evans & Sutherland. His prior work at MIPS includes having been principal architect of the Smart MIPS(tm) extensions for smart cards. He holds a degree in computer science from the University of California at Berkeley.

Kittur Ganesh

Kittur Ganesh is a Sr.Technical Consulting Engineer at Intel providing consulting, support and training on various software products targeting Intel® Architecture. Previously, for 6+ years at Intel, Kittur designed and developed software primarily used for fracturing design data of Intel chips. Prior to joining Intel, Kittur was involved in developing commercial SW in the EDA industry for 10+ years. Kittur has a M.S (Computer Science), M.S (Industrial Eng.) and a B.S (Mechanical Eng.)

Klaas de Waal

Klaas de Waal has been writing software for systems ranging from Seiko 8-bit microprocessors to Silicon Graphics workstations. He was writing applications for Linux at the time when it was still called Unix, AIX, Irix or Solaris. Instead of writing software he now spends most of his time porting existing open-source applications to the Philips Semiconductor's Nexperia platform and talking about it.

Liam Girdwood

Liam Girdwood is a Software Engineer at Wolfson Microelectronics. He specialises in embedded Linux audio and, prior to joining Wolfson, was an embedded Linux consultant with HP (now Agilent). He has worked within the Free Software community for more than five years. Liam Girdwood holds a Computer Science Degree from Heriot-Watt University. In his spare time, he is the maintainer of several open source projects including libnova (http://libnova.sf.net), a celestial mechanics library used by robotic telescopes around the world.

Manas Saksena

Manas Saksena is the CTO of Time Sys, where he is responsible for the technology strategy and development for Embedded Linux products and services. He is also the chair of the real-time working group at CELF. In his past life, he spent a few years doing research in real-time systems, and published a number of papers on that topic.

Marcin Klecha

Mark Gross

Mathew Locke

Mathieu Desnoyers

Mathieu Desnoyers is interested in kernel programming internals and computer security. He contributed to the rt2570 wireless driver, to the Linux Kernel time subsystem and he took part in technical discussions with the System TAP team. He is active in the 2600 Montr�al group, follows security conferences and competitions.

Since November 2005, he has become the maintainer of the Linux Trace Toolkit (LTT) project, taking over the development with the new LTTng. He is the author of Linux Trace Toolkit Next Generation (LTTng) project which started in may 2005 (http://ltt.polymtl.ca). He is the main developer of Linux Trace Toolkit Viewer (LTTV) since the project started in 2003. He is currently completing a M.Sc.A. in Computer Engineering at Ecole Polytechnique de Montr�al.

Matthew Klahn

Matthew Locke

Matt Mackall

Min-Seok Jang

Munehiro Ikeda

Nicholas Mc Guire

Rob Landley

Rob Landley is a Senior Linux Engineer at Time Sys Corporation and the new maintainer of BusyBox. He first encountered Linux when the SLS disks came across Fidonet in 1993, and has been building various Linux systems from source code for six years. He co-founded a combination Linux Expo and Science Fiction convention (Penguicon) because that really is his idea of fun, as are writing documentation, researching computer history, urban hiking, and knitting chain mail.

Ruud Derwig

Scott Preece

Scott Preece is a senior software architect at Motorola Mobile Devices, working in Champaign-Urbana, Illinois, on developing a mobile-phone platform based on Linux. He has been working in software development close to 40 years, with extended periods in information retrieval research, user interaction design, UNIX system development, LISP compilers, and mobile-phone software systems. He is a Motorola Science Advisory Board Associate and was chairman of Motorola's Software Engineering Technology Steering Committee.

Scott has a Computer Science PhD from the University of Illinois as well as degrees from Dartmouth College and the University of Chicago, dating to before Computer Science was a major.

Scott chairs the CELF Mobile Phone Profile Working Group and is a member of the CELF Architecture Group.

Shinichi Ochiai

Steve Johnson

Steve Johnson has been a Senior Software Engineer for a Panasonic R & D lab in Princeton, NJ for the last 14 years. He is currently the chair of CELF's Security Working Group. He graduated from the University of California, Davis in 1968, 1970, and 1977 from which you can draw two conclusions: 1) he's getting up there in years, and 2) you shouldn't make any assumptions about the sequence of years. In his spare time he likes to make and/or fix things, which pretty much describes his working day also. His current wish is to learn more Japanese words without forgetting the words learned last week.

Takanari Hayama

Takanari Hayama, Ph.D. is a founder and president of IGEL Co.,Ltd, an IT consulting company. He has been consluting Linux and open source solutions for more than 10 years. He has been supporting many kernel related development. He has developped Linux drivers for MPEG-2 Decoders, Digital TV related products, Professional Audio Device and so on. He also has involved in IPv6 development that runs not on only Linux, but also on NetBSD and ITRON. His current interrests are in the area of multimedia computing, operating system and mobile computing.

Thomas Gleixner

Thomas Gleixner has worked on industrial embedded devices for over 20 years. During the last 8 years he has focused on Linux for industrial applications. He has contributed to Linux in several ways. He is the main author of the hrtimer subsystem and the high resolution timer implementation on top of hrtimers and a major contributor to Ingo Molnars realtime preemption patch.

Todd Poynor

Todd works on embedded Linux OS features for Monta Vista Software, including development of power management technology for mobile devices and consulting with consumer electronics manufacturers on power management architecture.

Tohru Nojiri

Categories:

- Events
- ELC

- 2006

From: eLinux.org

ELC 2006 Presentations

Presenters, Demo-ers, Panelists and Participants: Thanks very much for your participation in CELF's Embedded Linux Conference I really enjoyed the conference, and hope you did as well.

Presenters: Please post your technical conference presentations on this page.

(See Instructions below the table)

Here is an article on LinuxDevices with images of the demo posters: http://www.linuxdevices.com/articles/AT8247255296.html

Table

Person	Session Description	Presentation
Tim Bird (Sony)	Keynote: 13 Years of Linux	Media:13-years-of-Linux.pdf
Andre Kruetzfeldt, Christophe Guinet	The MPPWG Mobile Phone Telephony API – part 2 Introduction to the Mobile Phone API	Media:ELC-Aplix-NEC.pdf
Armin Gerritsen	The Video Clip Player – Philips Nexperia™ PNX0106 and Linux based platform	Media:CELF-ELC_VideoClipPlayer.pdf
Chang-Sik Cho	Digital Entertainment Center Solution	Media:ELC-presentation-DEC-0412.pdf
Denis Kropp	Graphics Subsystem in an Embedded World - Integrating DirectFB into a UHAPI platform	Media:ELC 2006 - GraphicsSubsystemInAnEmbeddedWorld.pdf
Ed Plowman	Khronos Media API Update	Media:Khronos-API-CELF-Apr06.pdf
Greg Kroah Hartman	Tutorial: Write a real, working Linux driver	Media:celf_2006_tutorial.pdf Media:USB_tutorial_example_code.tar.gz
Greg Ungerer	uClinux -- Micro-controller Linux	Media:uclinux.pdf
Hyok S. Choi	Non-Paged Memory Management on Mainline ARM Kernel	.
Jared Hulbert	Tutorial: Creating optimized XIP systems	Media:CELF-XIP_Linux.pdf
John Goodacre	ARM MPCore and Power Management	Media:MPCore_and_Linux_Power.pdf, Media:MPCore_and_Linux_Power.ppt
Jordan Crouse	Taking the plunge - the marriage of X86 and embedded Linux	Media:jordan_crouse_celf_2006.pdf
Jung-Hyun Yoo	Linux on a Terrestrial DMB TV Receiver	.
Kevin D. Kissell	Microthreads as Linux CPUs - SMTC Linux for MIPS MT cores	Media:CELF_SMTC_April_2006_v0.3.pdf
Kittur Ganesh	Optimization Techniques for maximizing application performance on Multi-core processors.	Media:Ganesh-CELF.pdf, Media:Ganesh-CELF.ppt

Klaas de Waal	Myth TV on Philips Nexperia™ PNX8550 and Linux based platform	Media:MythTVonNexperia-CELF-05.pdf
Liam Girdwood	Enhancing ALSA audio for portable devices.	.
Manas Saksena	State of Linux Realtime – BOF	.
Mark Gross	Power Management Panel	.
Mathew Locke	CE Linux Forum Open Test Lab	.
Mathieu Desnoyers	Low disturbance embedded system tracing with Linux Trace Toolkit Next Generation.	Media:celf2006-desnoyers.pdf
Matt Mackall	Kernel Size Issues	.
Matthew Klahn	Visualizing resource usage during initialization of embedded systems	Media:VisualizingResUsageDuringBoot.pdf
Min-Seok Jang	Research on Linux 2.6 Kernel Features for CE products	.
Munehiro Ikeda	Examining Linux Kernel Size	Media:size_exam_celf_elc2006.pdf
Prof. Nicholas Mc Guire	Boot-Time Optimization - results of applying currently available solutions	Media:boot_opt.pdf
Rob Landley	What's new with BusyBox	.
Ruud Derwig	Audio Video Graphics Version 2.0 Specification	Media:CELF_AVG_Specification_v2_20060411.pdf (Media:CELF_AVG_Specification_v2_20060411.ppt)
Scott Preece	The MPPWG Mobile Phone Telephony API – part 1 Developing the API	Media:MppApiSession1.pdf
Shinichi Ochiai	Experience with realtime performance of the current Linux technologies	Media:ExperienceWithRealtimePerformance.pdf
Steve Johnson	Trusted Bootloader	Media:Trusted_Boot_Loader.pdf
Takanari Hayama	Analysis of User Level Device Driver usability in embedded application – Technique to achieve good real-time performance	Media:uldd060411celfelc2006.pdf
Thomas Gleixner	A Case Study of transitioning a project from RTAI to RT-Preempt	.
Todd Poynor	Topics in Embedded Power Management	Media:pm-celf-summit-2006.pdf
Tohru Nojiri	Kprobes implementation for Embedded System	.

Instructions for Presenters

Here are the steps to follow to add your presentation to this page (Please read them BEFORE clicking):

- If you don't have a wiki account, please create one by clicking on User Preferences, fill out the form, and return here.
  - (A wiki account is required to attach files or edit this page)
- PDF format is preferred. If you can convert your presentation to PDF, please do so.
- If your presentation has any message which says it is confidential (such as in a footer from a corporate or CELF template), please remove it.

    - If you used a CELF template that has "CELF confidential" in the footer, please upload the presentation so we can remove the footer. Or you can remove it yourself by editing the footer on the slide "master" page.
- Please make sure your filename does not have any spaces in it
- Click on the AttachFile link at the bottom of the page
    - on the form, enter your presentation file name (or browse to it)
    - click on "Add link to page".
    - click on "Upload" (your file will be added as an attachment to the page)
    - click on "ShowText" to see the page again.

If you want to be extra nice, you can add the appropriate "Media:" line to your entry in the table. Do this by clicking on EditText at the bottom of the page and copying the text from the bottom of the page to your line (where it currently has a ".").

See the entry for Tim Bird (first in the table) for an example of the syntax required. Each table line is one long line, even though it may wrap in your browser (so remove all line feeds from your line before saving this page.) Don't worry - if you don't do this or mess something up I will come along later and fix up the table for you.

Thanks

Attachments

Below this line is a list of attachments:

- Media:13-years-of-Linux.pdf
- Media:celf2006-desnoyers.pdf
- Media:MppApiSession1.pdf
- Media:pm-celf-summit-2006.pdf
- Media:Ganesh-CELF.pdf
- Media:Ganesh-CELF.ppt
- Media:ELC2006-GraphicsSubsystemInAnEmbeddedWorld.pdf
- Media:VisualizingResUsageDuringBoot.pdf
- Media:CELF_SMTC_April_2006_v0.3.pdf
- Media:celf_2006_tutorial.pdf
- Media:USB_tutorial_example_code.tar.gz
- Media:Trusted_Boot_Loader.pdf
- Media:ExperienceWithRealtimePerformance.pdf
- Media:size_exam_celf_elc2006.pdf
- Media:CELF_AVG_Specification_v2_20060411.pdf
- Media:CELF_AVG_Specification_v2_20060411.ppt
- Media:uclinux.pdf
- Media:MythTVonNexperia-CELF-05.pdf
- Media:CELF-ELC_VideoClipPlayer.pdf
- Media:uldd060411celfelc2006.pdf
- Media:MPCore_and_Linux_Power.pdf
- Media:MPCore_and_Linux_Power.ppt
- Media:ELC-Aplix-NEC.pdf
- Media:ELC-presentation-DEC-0412.pdf
- Media:Khronos-API-CELF-Apr06.pdf
- Media:boot_opt.pdf

Categories:

- Events

- ELC
- 2006
- Presentations

> From: eLinux.org

ELC 2007 Call For Presentations

The CE Linux Forum would like to invite you to make a presentation at our upcoming Embedded Linux Conference. The conference will be held April 17, 18 and 19 in San Jose, California.

(*See Embedded Linux Conference 2007 for general information* about the conference.)

Contents

Presentations Page

- All presentations can now be found on ELC_2007_Presentations.

Guidelines

Presentations should be of a technical nature, covering topics related to use of Linux in embedded systems. The CE Linux Forum is focused on the use of Linux in consumer electronics products, but presentations may cover use of Linux in other embedded areas, as long as the topic is of general relevance to most embedded users.

Presentations that are commercial advertisements or sales pitches are not appropriate for this conference.

Presentations on the following topics are encouraged:

- Audio, Video, and Graphics systems for embedded products
- Security
- System size
- Bootup time
- Meeting real-time constraints
- Power management
- Streaming media
- Flash memory devices and filesystems
- Technologies related to cell phones, digital settop boxes, handheld devices, or other CE products
- Development tools for embedded users
- Use of Linux in actual products, practical experience and war stories
- Standards for CE products

Most presentation slots are 50 minutes long, including time for questions. A few shorter 40-minute slots are also available.

A paper submission is NOT required in conjunction with the presentation. However, if a paper is produced, it is requested to be published in wiki format on the Embedded Linux Wiki.

You may also submit a request to present a tutorial. Tutorials are intended to be longer, interactive learning sessions. A tutorial may occupy more than one session slot at the conference.

We prefer presentations that have not been previously presented, but exceptions will be made for material that is either of great interest or has been seen previously only by a limited audience.

A single person may submit more than one proposal. However, since the number of slots is limited, it is likely that at most one proposal from an individual will be accepted.

For examples of presentations from previous years' conferences, see: Tech Conference 2005Docs and ELC 2006 Presentations

Technical Facilities

Presentations will be in rooms with 1024�768 XGA data projectors. Please bring your own laptop for your presentation. Please contact us if you have further requirements.

Reward and possible assistance

Speakers will receive complimentary professional registration for the conference. That is, if you make a presentation, you get into the rest of the conference for free

Travel and accomodation assistance is available for a limited number of speakers, under certain circumstances. Please contact the e-mail address below if you believe your circumstances would qualify you for this assistance.

Review

All presentation, tutorial, demo and BOF proposals will be reviewed by the CE Linux Forum, and submitters will be notified of their acceptance (or not) by March 1, 2007.

Proposals will be assessed on a number of criteria:

- Is the topic of interest to embedded Linux developers? That is, is it relevant for using Linux in embedded products?
- Does the presentation describe something new or interesting?
- Does the topic fit with other presentations planned for the conference?
- Is source code for the project (if applicable) available now?
- May the forum publish your presentation on it's web site?

Submission method

To submit a presentation proposal, send an e-mail to celf-elc@tree.celinuxforum.org

A presentation request should include your name, a presentation title and a brief (one or two paragraph) abstract. Also, please indicate with your proposal whether your presentation may be published by the forum on it's web site, after the conference.

If your proposal is accepted, you will be requested to provide a short paragraph describing yourself, for use in conference materials.

If you have questions, you may also send queries to the e-mail address above. We will be happy to answer questions about the presentation requirements or the conference itself.

Deadlines

Proposals for presentations, demos and Birds-of-a-Feather sessions must be received by February 5, 2007.

Speakers will be notified by March 1, 2007.

Categories:

- Events
- 2007
- ELC

ELC 2007 Presentations

Presentations

Session Description	Person	Presentation
Keynote: Embedded Linux - An Increasing Nightmare?	Thomas Gleixner	Celf-2007-keynote-Gleixner.pdf LWN.net writeup and comments
Keynote: The State of the Linux Kernel	Jonathan Corbet	corbet-kernel.pdf or LWN page.
X (Not On The Desktop)	Matthew Allum	http://folks.o-hand.com/mallum/presentations/x-not-on-the-desktop/slide-00.html
The use of JTAG in Linux Bring-up	Mike Anderson	JTAG-Anderson.pdf
Kernel Probes for ARM	Quentin Barnes	kprobes_for_ARM-ELC2007.pdf
Kernel Probes for MIPS, ARM and PPC32	Tim Bird	Kernel_Probes_for_MIPS_ARM_and_PPC32-ELC2007.pdf
How to Participate in the Kernel Development Process	Jonathan Corbet	corbet-dev-process.pdf or LWN page
TimeDoctor – Use the Strength of Eclipse to Visualize (Multi)Processor Execution Behavior	Ruud Derwig	TimeDoctor_ELC_20070417.pdf
Audio, Video and Graphics BOF	Ruud Derwig	not available
Benchmarking of Dynamic Power Management Solutions	Frank Dols	Benchmarking-of-Dynamic-Power-Management-Solutions.pdf
SPlit Application architeCturE	Bas Engel	not available
The Current Status of Timers and Realtime Support in the Kernel	Thomas Gleixner	http://tglx.de/private/tglx/celf2007/celf-2007-rt.pdf
Power Management Techniques, Policies, and Problems for Embedded Linux	Mark Gross	CELF_ELC2007_mgross_PM_slide_set.pdf
Power Management BOF	Mark Gross	not available
TomoyoLinux - A Lightweight and Manageable Security System for PC and Embedded Linux	ToshiharuHarada, Tetsuo Handa	elc2007-presentation-20070418.pdf or elc2007-presentation-20070418-for_linux.pdf (should work with most PDF readers)
How DirectFB Adopted Market Specific Requirements	Takanari Hayama	celf2007_directfb.pdf
Current State of Bluetooth Support in Linux	Marcel Holtman	elc2007_slides.pdf
System size BOF	Jared Hulbert	not available
Realtime BOF - Realtime Preempt Patch Adaptation Experience (including Commercial Product)	YungJoon Jung	RT-BoF-2007-04-17.pdf
FancyPants – An Advanced 2D Graphics System for CE Linux	Robi Karp	fst-fancypants-celf2007.pdf - More information at: http://www.fancypants-graphics.com/
Bootup Technologies BOF	Elias Kesh	BTWG-Discussion-Plenary2007.pdf
OpenEmbedded - Easy QA,		

Repeatability and Retargeting		
Prelinker Usage for MIPS Cores	Arvind Kumar, Kazu Hirata, Shinichi Tsurumoto	Evaluation_of_MIPS_Prelinking.pdf
How To Protect Your Intellectual Property While Using Open Source	Shawn Kwon	CELF-ELC07-OSS.pdf
Applying User-level Drivers on DTV System	Gunho Lee	UDD_on_DTV_ELC2007.pdf
Analysis of Interrupt Entry Latency in Linux 2.4 vs 2.6	SangBae Lee	CELF_ELC_Interrupt_Latency_2.4_vs_2.6.pdf
A Generic Parameter Layer for Linux Power Management	Matt Locke	mlocke-elc2007-pm.ppt.pdf
SPE-assisted User Level Device Driver on Cell Processor	Hiroyuki Machida	20070419-Cell-Cloop-e.pdf
System-wide Memory Profiling	Matt Mackall	memory-profiling.html
TomoyoLinux – Tutorial	Kei Masumoto, Kentaro Takeda	PDF on Sourceforge
Porting and Evaluating the Linux Realtime Preemption on Embedded Platform	Katsuya Matsubara	preempt070418celfelc.pdf preempt070418celfelc.odp
Kernel Debugging with GDB	Nicholas McGuire	not available
Kernel Validation Tools	Nicholas McGuire	tools_slides.pdf
GDB Tracepoints for GNU/Linux User and Kernel Space	Nicholas McGuire	not available
Telepathy: Real-time Communications Framework	Robert McQueen	telepathy-elc2007.pdf
Comparison of Secure OSes and embedded SELinux activity in Japan	Yuichi Nakamura	SecureOS_nakamura.pdf
Suspend-to-RAM implementation on freescale 74xx without PMU	Fujihito Numano	Suspend-to-RAMImplementationOnFreescale74xxWithoutPMU-070418.pdf
Mobile Convergence Computing Handset Supporting Ubiquitous Concept	Tae Joon Park	Mobile_Convergence_Communicator_ELC2007.pdf
OpenKODE - The Khronos Open Development Environment	Ed Plowman	not available
CELF in the Mobile Phone Space	Scott Preece	CelfInMobilePhoneSpace.pdf
Mobile Phone BOF	Scott Preece	not available
Experiment with Linux and ARM Thumb-2 ISA	Philippe Robin	Experiment_with_Linux_and_ARM_Thumb-2_ISA.pdf
File System Survey, Focus on Embedded Linux	Gene Sally	filesystems-for-embedded-linux.pdf
Fitting Linux on Resource Constrained Targets	Gene Sally	linux-on-space-constrained-systems.pdf
Gstreamer Tutorial	Jan Schmidt	celinux-gst-tutorial.pdf gst-code-samples.tar.gz
HTTP-FUSE PS3 Linux which is internet boot framework with kboot	Toshiki Yagi	elc07_yagi.pdf
uClinux -- State of the Nation	Greg Ungerer	uclinux-sotn.pdf

uClinux -- State of the Nation	Greg Ungerer	uclinux-sotn.pdf
The OpenMAX Integration Layer standard	Giulio Urlini	The_OpenMAX_Integration_Layer_standard.pdf
The CEA 2014 Standard. A New XHTML-Based Browser and Setup Framework for Digital Home Devices	Mark R. Walker	CEA_2014_Overview.pwz
An Annoucement from the GNOME Foundation	Jeff Waugh	See the LWN.net writeup.
Deferred Dynamic Loading --- A Memory Reduction Technique	Tetsuji Yamamoto	DeferredDynamicLoading_20070417.pdf
Management of Software Suspend Image	Haitao Zhang	not available

Categories:

- Events
- 2007
- ELC

ELC 2008 Presentations

Presentations

- Video of the ELC 2008 keynotes and various sessions can now be viewed online at http://free-electrons.com/community/videos/conferences.
- An extensive report of this year's conference is now online at http://free-electrons.com/articles/conferences/elc2008-report.

Keynotes

Person	Session Description	Presentation
Henry Kingman	Tux in Lights	henry_kingman_welcoming_address.pdf
Andrew Morton	The Relationship Between kernel.org Development and the Use of Linux for Embedded Applications	morton-elc-08.ppt
Tim Bird	Status of Embedded Linux and CELF Plenary Meeting	Status-of-embedded-Linux-ELC2008.ppt

Sessions

Person	Session Description	Presentation
Kate Alhola	Maemo Mobile Linux Platform, Current Status and Future Directions	elc_maemo_2008.pdf
Mike Anderson	Using a JTAG for Linux Driver Debugging	CELF_JTAG_Anderson.ppt
Alexander Belyakov	Compressed Swap Solution for Embedded Linux	belyakov_elc2008_compressed_swap_final_ppt.pdf belyakov_elc2008_compressed_swap_final_doc.pdf
Hugh Blemings	Learning Kernel Hacking from clever people	elc-us-2008-slides-final.pdf
Andrew Christian	Compiling Full Desktop Distributions for ARM: The Handhelds Rebuild Project	HandheldsMojo_ELC2008.pdf
Felipe Contreras	Gstreamer and OpenMAX IL: plug and play	gst-openmax.pdf
Jake Edge	Avoiding Web Application Flaws in Embedded Devices	LWN Article
Jörn Engel	Status of LogFS	not available
Nils Faerber	GPE Phone Edition - An Open Source Software Stack for Linux Mobile Phones	gpe2-celf2008.odp
Klaas van Gend	Using Real-Time Linux	Using_Real-Time_Linux.KlaasVanGend.ELC2008.pdf
Liam Girdwood	Every Microamp is Sacred - A Dynamic Voltage and Current Control Interface for the Linux Kernel	regulator-api-celf.pdf

Mark Gross	Power Management Quality of Service and How You Could Use it in Your Embedded Application	elc2008_pm_qos_slides.pdf
Takanari Hayama	DirectFB Internals - Things You Need to Know to Write Your DirectFB gfxdrive	elc2008_directfb_gfx.pdf
Seo Hee	APCS (ARM Procedure Call Standard) Tutorial	CELF_APCS_Seohee_lge.pdf
Seo Hee	Trouble Shooting for Blocking Problem	not available
Kevin Hilman	Building Blocks for Embedded Power Management	PM_Building_Blocks1.pdf
Stuart Hughes	Roll-Your-Own Linux the Easy Way with LTIB	celf_ltib_bof_v1 1. pdf
Jared Hulbert	AXFS: Architecture and Results	AXFS_at_ELC_2008.ppt
Hirohisa Iijima	Episodes of LKST for Embedded Linux Systems	EpisodesLKST_Lineo_CELF_ELC2008.pdf
YoungJun Jang	Avoiding OOM on Embedded Linux	CELF_AvoidOOM.pdf
YungJoon Jung	Real-Time Linux BOF	RT-BoF-2008-04-15.pdf
Jong-Sung Kim	Back-tracing in MIPS-based Linux Systems	ELC2008 - Back-tracing in MIPS-based Linux Systems.pdf
Min-Chan Kim, Oleksiy Kokachev	Instant Startup for Application Using Reducing Relocation Time and Rearrange Function	DDLink FunctionReorder 08 04.pdf
KaiGai Kohei	Recent Security Features and Issues in Embedded Systems	ELC2008_KaiGai.pdf
Jyunji Kondo	Development of Mobile Linux Open Platform	Development_of_Mobile_Linux_Open_Platform.pdf
Rob Landley	Cross Compiling Linux (tutorial)	not available
Grant Likely	Shifting Sands: Lessons Learned from Linux on an FPGA	glikely--fpga-lessons-learned.pdf
Grant Likely	A Symphony of Flavours; Using the Device Tree to Describe Embedded Hardware	glikely--device-tree.pdf
Matt Locke	Building Custom Embedded Linux Distributions	mlocke-elc2008-oe.pdf
Matt Mackall	Kernel Size Report, and Bloatwatch Update	elc2008.odp
Guido Madaus	Disko - An Application Framework for Digital Media Devices	elc.tar
David Mandala	UME - Ubuntu Mobile and Embedded	UbuntuMobileEmbedded.pdf
Katsuya Matsubara, Hisao Munakata	Using UIO on an Embedded Platform	uio080417celfelc08.pdf
Nicholas McGuire	Real-Time Virtualization Solutions for Linux - A Comparison of Strategies	not available
Yuichi Nakamura	Development of Embedded SE Linux	ELC2008_nakamura.pdf

Jeff Osier-Mixon	Effectively Managing Documentation for Embedded Linux Projects	jeffrey-osier-mixon-elc.ppt
Kyungmin Park, Sunmi Yoo	Filesystem Support on Multi Level Cell (MLC) Flash in Open Source	ELC2008 Filesystem support on Multi Level Cell flash in open source.ppt
Thomas Petazzoni	Linux Tiny - Penguin Weight Watchers	linux-tiny.pdf
Matt Porter	Leveraging Free and Open Source Software in a Product Development Environment	elc-foss.pdf
Conrad Roeber	Enhancements to USB Gadget Framework	ELC2008-gadget-enhancements-web.pdf
Frank Rowand	Adventures In Real-Time Performance Tuning	mips_real_time.pdf
Gene Sally	How GCC Works, An Embedded Engineer's Perspective	GCC_Tips
Deepak Saxena	Appropriate Community Practices: Social and Technical Advice	not available
Christian Schaller	GStreamer on Embedded - Latest Developments and Features	celinux-mountainview-gstreamer-tim.ppt
Michael Shiloh	OpenMoko	not available
York Sun	Adding Framebuffer support for Freescale SoCs	Adding Framebuffer support to Freescale SoCs York Sun.ppt
Kentaro Takeda	How to Analyze Your Linux's Behavior with TOMOYO Linux	elc2008.pdf
JT Thomas	Embedded Linux Development with Eclipse	Embedded Linux Development with Eclipse.ppt
Justin Treon	Making a Phone Call With Phase Change Memory	Making_a_Phone_Call_With_PCM_at_ELC_2008.ppt
Richard Woodruff	Linux System Power Management on OMAP3430	TI_OMAP3430_Linux_PM_reference.ppt

Categories:

- Presentations
- ELC
- 2008
- Events

From: eLinux.org

ELC 2009 Presentations

Contents

Intro

Presenters, Demo-ers, Participants: Thanks very much for your participation in CELF's Embedded Linux Conference 2009.

This page is for collecting the presentations that were made at the conference. During and after the conference we will collect materials from the presenters and place them here. Please watch this page if you are interested in a particular presentation - and it if doesn't show up, please send me an e-mail and we'll try to track it down.

Videos

Video from the conference can be found here courtesy of Free Electrons: Video Presentations

Free Electrons has been kind enough to provide some of these presentations in High Definition, making it possible to more easily read slides directly from the video.

Instructions

Presenters: Please post your technical conference presentations on this page. (See Instructions below the tables)

Table of Presentations

Keynotes and Panel

Presenter(s)	Session Description	Presentation
Dirk Hohndel	Ubiquitous Linux	No slides - but see ELC2009: Ubiquitous Linux (LWN.net)
David Woodhouse	Embedded Linux and Mainline Kernel	dwmw2-ELC-2009-04.pdf dwmw2-ELC-2009-04.odp
Tim Bird (moderator)	Embedded Linux Kernel Features and Development Panel	No slides - but see ELC/LFCS2009 A tale of two panels (LWN.net)

Presentations

Presenter(s)	Session Description	Presentation
Kate Alhola	Maemo 5 (Fremantle), mobile Linux platform with cellular connectivity	fremantle_elc_2009.pdf
	Animated UI technologies in	

Kate Alhola	Maemo 5 (Fremantle), mobile Linux environment	animated_ui_elc_2009.pdf See also Kate's blog entry for ELC
Mike Anderson (presented by Reece Pollack)	User-Space, Multi-core Development Issues	UserSpace_Multicore-Slides_Anderson.pdf
Mike Anderson (presented by Reece Pollack)	What are Interrupt Threads and How Do They Work?	InterruptThreads-Slides_Anderson.pdf
Jeff Arnold	Ksplice: Rebootless kernel updates	elc2009-ksplice.pdf
Eric Cloninger	Building an Embedded Tools Standard Using Eclipse	elc2009-building_an_embedded_tools_standard_using_eclipse.pdf
Magnus Damm	Runtime Power Management on SuperH Mobile	Runtime-Power-Management-on-SuperH-Mobile-20090407.pdf
David Daney	Some new tricks for better performance in MIPS-Linux	new-tricks-mips-linux.pdf
Mathieu Desnoyers	Deploying LTTng on Exotic Embedded Architectures	LTTng-presentation-celf-2009-0.2.pdf desnoyers-celf2009-paper.pdf
Anna Dushistova, Alexandre Rusev and John Mehaffey	Debugging with JTAG	DebuggingWithJtagCelf2009.pdf
Jake Edge	Security issues for embedded devices	security-issues.pdf LWN page
Klaas van Gend	Top 3 pains in professional use of bitbake	ELC.klaasvangend.openembedded.v4.pdf
Toru Homma	Evaluation of Flash File Systems for Large NAND Flash Memory	ELC2009-FlashFS-Toshiba.pdf
Edgar E. Iglesias	Debugging and profiling embedded Linux/CRIS systems with QEMU	elc2009-qemu-cris.pdf
Jaehoon Jeong	Dynamic Instrumentation of user-space application based on kprobe	ELC2009_User_space_dynamic_instrumentation_based_on_kprobe-0331-final.pdf
Bhagyashri Hemant Katole	Embedding Network Devices with Linux	(session was cancelled)
Dongsoo Kim, HeungJun Kim	Framework for digital camera in Linux	Framework_for_digital_camera_in_linux-in_detail.ppt
Denis Oliver Kropp	DirectFB II	(session was cancelled)
Grant Likely	It's Alive! - Linux on Embedded PowerPC porting guide	glikely-powerpc-porting-guide.pdf
	Tux Meets Radar	

Grant Likely	O'Reilly - Linux in Military Telecom	oreilly.pdf
Bruno Cardoso Lopes	Understanding and writing an LLVM Compiler Backend	LLVM-ELC2009.pdf
Matt Mackall	Visualizing Process Memory	smem.pdf // In case color mapping error observed, try this file ELC2009PresentationsM.pdf
Dan Malek	Memory...The Most Precious Resource	celf_mem_notify.pdf
David Mandala	Ubuntu ARM Distribution	UbuntuARM.pdf
William Marone	Distributed Cross Platform Test Automation	DistributedCrossPlatformTest.pdf
Paul Mundt	Superpages Revisited: Transparent Application of Large TLBs on Embedded Systems	elc2009-superpages.ppt
Michael Opdenacker	Update on filesystems for flash storage	flash-filesystems.pdf
Jeffrey Osier-Mixon	Cooperative Development Inside Communities	CommunityDevelopment.pdf
Conrad Parker	A Linux multimedia framework for SH-Mobile processors	article elc-shmobile-multimedia.pdf
Rodolph Perfetta	The Web in your Hand - Optimizing Browsing Experience with ARM Embedded Linux Devices	CELF.pdf
Thomas Petazzoni	Building Embedded Linux Systems with Buildroot	buildroot.pdf
Matthew Porter	Video4Linux: What about Output?	elc09_mattporter_v4l.pdf
Andre Puschmann	Quantitative analysis of system initialization in embedded Linux systems	ELC09_boottime_reduction.pdf
Jim Ready	Plan Your Work, Work Your Plan: Avoiding Common Linux Development Stumbling Blocks	.
Frank Rowand	Musings on analysis of measurements of a real-time workload.	musings_on_analysis_of_measurements_of_a_real-time_workload.pdf
Leandro Melo de Sales	BRisa UPnP Framework for Embedded Systems	brisa_ELC.pdf
Christian F.K. Schaller	Basic video editing on embedded devices using GStreamer	celinux-sanfran-gstreamer.pdf celinux-sanfran-gstreamer.ppt

Madhvesh Sulibhavi (presented by Tim Bird)	KProbes and Systemtap Status	Kprobes-Systemtap-Status-from-Sony-for-ELC09.pdf
John Williams	Embedded Linux on FPGAs for fun and profit	ELC2009_Embedded_Linux_on_FPGAs_for_fun_and_profit.pdf

Birds-of-a-Feather Sessions

Presenter(s)	**Session Description**	**Presentation**
Thomas Petazzoni	Build Tools	building-tools-bof.pdf
Tim Bird	eLinux Wiki	eLinux-wiki-BOF-ELC-2009.pdf
Matt Locke	Embedded Security	ELC-2009-Security-BoF-mlocke.pdf
Michael Opdenacker	System Size	size-bof.pdf

Categories:

- Presentations
- ELC
- 2009

> From: eLinux.org

ELC 2010 Call for Presentations

The CE Linux Forum would like to invite you to make a presentation at our upcoming Embedded Linux Conference. The conference will be held April 12-14, 2010 in San Francisco, California.

Please see the embeddedlinuxconference.com ELC 2010 web site for more information about the conference.

Contents

Guidelines

Presentations should be of a technical nature, covering topics related to use of Linux in embedded systems. The CE Linux Forum is focused on the use of Linux in consumer electronics products, but presentations may cover use of Linux in other embedded areas, as long as the topic is of general relevance to most embedded users.

Presentations that are commercial advertisements or sales pitches are not appropriate for this conference.

Presentations on the following topics are encouraged:

- Audio, Video, and Graphics systems for embedded products
- Security
- System size
- Bootup time
- Meeting real-time constraints
- Power management
- Streaming media
- Flash memory devices and filesystems
- Build systems
- Embedded distributions
- Development tools for embedded users
- Technologies related to cell phones, digital settop boxes, televisions, cameras, handheld devices, or other CE products
- Use of Linux in actual products, practical experience and war stories
- Standards for CE products

Most presentation slots are 50 minutes long, including time for questions.

No paper required

A paper submission is NOT required in conjunction with the presentation. However, if a paper is produced, it is requested to be published in wiki format on the Embedded Linux Wiki

Tutorials

You may also submit a request to present a tutorial. Tutorials are intended to be longer, interactive learning sessions. A tutorial may occupy more than one session slot at the conference.

Birds of a Feather (BOF) sessions

You may also submit a request to be the session leader for a "Birds Of A Feather" (BOF) session. This is an informal discussion on a topic of group interest. For a BOF proposal, please describe the topic area you would like to discuss. You should come prepared to help move the discussion along with a "lead in" presentation (a few slides) or some thought-provoking questions.

Technical Showcase (Demo)

Finally there will be a "poster" demo session at the event. The demo session consists of table-top demos, with an accompanying poster describing each demo. In this session, the demonstrator can interact with individuals or small groups, describe their work and answer questions.

General Guidelines

We prefer presentations that have not been previously presented, but exceptions will be made for material that is either of great interest or has been seen previously only by a limited audience.

A single person may submit more than one proposal. However, since the number of slots is limited, it is likely that at most one proposal from an individual will be accepted.

Examples from previous years

For examples of presentations from previous years' conferences, see:

- ELC 2009 Presentations
- ELC 2008 Presentations
- ELC 2007 Presentations
- ELC 2006 Presentations

For information about previous demo sessions, see the ELC 2009 Technical Showcase page

Technical Facilities

Presentations will be in rooms with 1024×768 XGA data projectors. Please bring your own laptop for your presentation. Please contact us if you have further requirements.

Reward and possible assistance

Speakers for presentation and tutorial sessions, as well as leaders of BoF sessions, will receive complimentary professional registration for the conference. That is, if you make a presentation or lead a session, you get into the rest of the conference for free! Please note, that if a session has multiple speakers, only the primary speaker for that session will receive free admission.

Please note that event admission includes access to the Linux Foundation Collaboration Summit, which is normally an invitation-only event.

Travel and accomodation assistance is available for a limited number of speakers, under certain circumstances. Please contact the e-mail address below if you believe your circumstances would qualify you for this assistance.

Review Process

All presentation, tutorial, demo and BOF proposals will be reviewed by the ELC program committee, and submitters will be notified of their acceptance (or not) by February 5, 2010.

Proposals will be assessed on a number of criteria:

- Is the topic of interest to embedded Linux developers?
  - That is, is it relevant for using Linux in embedded products?
- Does the presentation describe something new or interesting?
- Does the topic fit with other presentations planned for the conference?
- Is source code for the project (if applicable) available now?
- May the forum publish your presentation on it's web site?

Submission method

To submit a presentation proposal, send an e-mail to elc10@tree.celinuxforum.org

A presentation proposal should include:

- the name of the presenter
- the presentation (or session) title
- a brief (one or two paragraph) description
- Also, please indicate with whether your presentation may be published by the forum on a web site, after the conference.

If you are submitting for someone else, please provide an e-mail address for the presenter.

If your proposal is accepted, you will be requested to provide a short paragraph describing yourself, for use in conference materials.

If you have any questions about the submission requirements, the status of your submission, the review process, or the event itself, please feel free to contact the program committee, at: elc10@tree.celinuxforum.org.

Deadlines

Proposals for presentations, demos and Birds-of-a-Feather sessions must be received by January 15, 2010.

Speakers will be notified by February 5, 2010.

Categories:

- Presentations
- ELC
- 2010
- Events

ELC 2010 Presentations

Presenters, Demo-ers, Participants: Thanks very much for your participation in CELF's Embedded Linux Conference 2010.

This page is for collecting the presentations that were made at the conference. During and after the conference we will collect materials from the presenters and place them here. Please watch this page if you are interested in a particular presentation - and it if doesn't show up, please send me an e-mail and we'll try to track it down.

Contents

Videos

Videos from the conference are available on http://free-electrons.com/blog/elc-2010-videos/

Instructions

Presenters: Please post your technical conference presentations on this page. (See Instructions below the tables)

Table of Presentations

Keynotes

Presenter(s)	Session Description	Presentation
Greg Kroah-Hartman	Android: A Case Study of an Embedded Linux Project	PDF | ODP | TGZ (with notes and license)
Matt Asay	Embedded in 2010: An End to the Entropy?	PDF

Presentations

Presenter(s)

Session Description

Presentation

Mike Anderson

Using a JTAG to Debug Linux Device Drivers

PDF

Mike Anderson

Using Interrupt Threads to Prioritize Interrupts

PDF

Mike Anderson

Creating a Secure Router Using SELinux

PDF

Mike Anderson

Strategies for Migrating Uniprocessor Code to Multi-Core SMP

PDF

Steve Bennett

Effective Use of Scripting in Embedded Devices

Slides | Paper

Tim Bird

State of Embedded Linux

PDF | ODP

Magnus Damm

Kexec - Ready for Embedded Linux?

PDF

Kevin Dankwardt

Effective Use of RT-Preempt

ODP

Lucas Martins De Marchi

Multi-core Scheduling Optimizations for Soft Real-time Multi-threaded Applications -- A Cooperation Aware Approach

PDF | ODP

Mathieu Desnoyers

Using the LTTng Tracer for System-wide Performance Analysis and Debugging (Hands-On Tutorial)

PDF | examples

Jake Edge

Understanding Threat Models for Embedded Devices

PDF | ODP

Mark Gross

Experiences in Android Porting, Lessons Learned,Tips and Tricks

PDF

Kevin Hilman

Runtime Power Management: Overview and Platform Implementation

PDF

YungJoon Jung and DongHyouk Lim

Measuring Responsiveness of Linux Kernel on Embedded System

PDF

Hiromasa Kanda

Lock-free Algorithm for Multi-Core Architecture

PDF

Jeremy Katz

An Introduction to the Qt Development Framework

PDF

Yoshitake Kobayashi

Evaluation of Data Reliability on Linux File Systems

PDF

Yong Bon Koo and Youngbin Seo

DVFS for Embedded Linux

PDF

Rob Landley

Developing for Non-x86 Targets Using QEMU

PDF

Melanie Rhianna Lewis

Case Study - Embedded linux in a Digital Television STB

PDF

Grant Likely

Flattened Device Tree ARM Support Update

PDF

Dan Malek

Embedded Multi-core with Adeos

German Monroy

Wake-ups Effect on Idle Power for Intel's Moorestown MID and Smartphone Platform

PDF

Jeff Osier-Mixon

Effectively Managing Documentation for Embedded Linux Projects

PDF

Jacob Pan

Porting the Linux Kernel to x86 MID Platforms

PDF

Steven Rostedt

Ftrace - Embedded Edition

ODP

Frank Rowand

Real-Time Linux Failure

PDF

Leandro Melo de Sales

Understanding and Developing Applications for Maemo Platform

PDF

Gene Sally

GPIO: Talking to the Outside World

ODP

David Schleef

Recent Developments in Open Video Technology

PDF

Frank Scholz

Mirabeau - Creating Personal Media Networks and Bridging DLNA/UPnP Devices Over The Internet

ODP, PDF

Masahiko Takahashi

A Consideration of Memory Saving by Efficient Mapping of Shared Libraries

PDF

Rob Taylor

Semantic Data Storage for Mobile Devices

PDF

Sujith Thomas

Workload-based Aggressive Power Management on the Intel Moorestown MID and Future Intel MID/Smartphone Platforms

PDF

Matthew Tippett

Engaging Developer Communities: Lessons and Opportunity from webOS

PDF

Dominique Toupin

Linux Toolchain Overview with Advanced Debugging and Tracing Features

PDF

Bill Traynor

eLinux.org wiki Present & Future

PDF

Greg Ungerer

Linux Without a Boot Loader?

PDF | ODP | source

Hans Verkuil

Supporting SoC video subsystems in video4linux

ODP

Denys Vlasenko

Link Time Dead Code and Data Elimination Using GNU Toolchain

ODP | PDF

Alexey Volkov

Implementing Asynchronous Zero-Copy API for Embedded IVR Application

Slides, Paper

David VomLehn

No Crash Dump? No Problem!

PDF

John Williams and Edgar Iglesias

Custom Hardware Modeling for FPGAs and Embedded Linux Platforms with QEMU

PDF

Vitaly Wool

Polishing Dirt: Porting RTOS Code to Linux Userspace Driver Framework

PDF | ODP

Benjamin Zores

GeeXboX Enna: embedded Media Center

PDF

Siji Sunny

Believable and fast Physics Based Animations on Box2D-Clutter

Birds-of-a-Feather Sessions

Presenter(s)	Session Description	Presentation
Grant Likely	Small Business Owners BOF	No slides
Bill Traynor	eLinux.org wiki Present & Future	PDF
David Mandala	Ubuntu on ARM	
Kevin Hillman	Power Management BOF	

Instructions for Presenters

Please create a link in the table for your presentation, copying the style of other links. (You may need to create an account in order to edit the wiki or upload files.)

When you have created the link, click on it to upload the file containing your slides.

Categories:

- ELC
- 2010
- Events
- Presentations

> From: eLinux.org

ELC 2011 Presentations

Presenters, Demo-ers, Participants: Thanks very much for your participation in Linux Foundation's Embedded Linux Conference 2011.

This page is for collecting the presentations that were made at the conference. During and after the conference we will collect materials from the presenters and place them here. Please watch this page if you are interested in a particular presentation - and if it doesn't show up, please send me an e-mail and we'll try to track it down.

Contents

Videos

Once again, through the diligent work of the Free Electrons team, all videos for ELC2011 can be found at ELC 2011 Videos and the Android Builder Summit videos can be found at Android Builder Summit videos.

Instructions

Presenters: Please post your technical conference presentations on this page. (See Instructions below the tables)

Table of Presentations

NOTE: If you add a wikilink to your presentation and attempt to upload it via the link, it may fail. If it does, use the Special:Upload page to upload your file.

Keynotes

Keynotes

Presenter(s)	Session Description	Presentation
Dirk Hohndel & Richard Purdie	The Yocto Project	No slides used.
Arnd Bergmann (IBM)	Becoming Part of the Linux Kernel Community	Elc2011_bergmann_keynote.pdf

Presenters

Presentations

Presenter(s)

Session Description

Presentation

Day 1, 10:00am

Keshava Munegowda (Texas Instruments)

Power Fail Safe FAT File Systems

Elc2011_munegowda.pdf

Frank Rowand (Sony)

Identifying Embedded Real-Time Latency Issues: I-Cache and Locks

Elc2011_rowand.pdf

Bruno Cardoso Lopes (University of Campinas)

LLVM, Clang and Embedded Linux Systems

Elc2011_lopes.pdf

Day 1, 11:00am

Steven Rostedt (Red Hat)

Kernel Shark Tutorial

Elc2011_rostedt.pdf

Kernelshark-tut-elc-2011.odp

Kang Dongwook (ETRI)

Snapshot Booting on Embedded Linux

Elc2011_kang.pdf

Khem Raj

State of OpenEmbedded Internal Toolchain and SDKs

Elc2011_raj_sdk.pdf

Day 1, 1:30pm

David Rusling (Linaro)

Linaro: A Year of Change

Linaro_2011_ELC_Talk.odp Linaro_2011_ELC_Talk.pdf

Hai Shalom (Atheros)

Control, Recover and Debug Your Embedded Product with PCD

Elc2011_shalom.odp

Gene Sally

Zigbee Networking & Linux

Day 1, 2:30pm

Xi Wang (Broadcom)

Solving Real-Time Scheduling Problems with RT_PREEMPT and Deadline-Based Scheduler

Elc2011_xi_rt.pdf

Mike Anderson (The PTR Group)

ARM Neon Instruction Set and Why You Should Care

Elc2011_anderson_arm.pdf

Darren Hart (Intel)

Yocto Project: Practical Kernel Development Tutorial

Day 1, 3:40pm

Arnd Bergmann (IBM)

Optimizations for Cheap Flash Media

Elc2011_bergmann.pdf

Wolfram Sang (Pengutronix)

Developer's Diary: Helping the Process

Elc2011_sang.pdf

Rajesh Lal (Nokia)

Fun with QML and JavaScript

Elc2011_lal.pdf

Day 1, 4:40pm

Thomas Gleixner (linutronix)

RT-Preempt: What's The State and Why There is No Roadmap

Elc2011_gleixner.pdf

Jason Kridner (Texas Instruments)

High-Level Web Interface to Low-Level Linux I/O on the Beagleboard

Elc2011_kridner.pdf

Day 2, 10:00am

Paul Mundt (Renesas)

Working with HardIRQs: Life Beyond Static IRQ Assignments

Elc2011_mundt.pdf

Amit Kucheria (Linaro)

Powerdebugging Inside Linaro

Elc2011_kucheria.odp Elc2011_kucheria.pdf

Mike Anderson (The PTR Group)

High-Performance Computing using GPUs

Elc2011_anderson_gpu.pdf

Day 2, 11:00am

Paul Larson (Linaro)

Linaro Automated Validation on ARM

ELC2011-Linaro-Validation.pdf

Dave Stewart (Intel)

The Yocto Project and its Application Development Toolkit (ADT) - The Answer to Effective Embedded Application Development

ELC_Yocto_ADT_2011_davest.odp

Damian Hobson Garcia (Igel), Katusya Matsubara, Takanari Hayama, Hisao Munakata

Integrating a Hardware Video Codec into Android Stagefright using OpenMAX IL

Elc2011_garcia.pdf

Day 2, 1:30pm

Koen Kooi (Texas Instruments)

Integrating OpenEmbedded and Yocto

Elc2011_kooi.pdf

Mark Gross (Intel)

How to Power Tune a Device Running on a Linux Kernel for Better Suspend Battery Life

Elc2011_gross.pdf

Remi Lorriaux (Adeneo Embedded)

Real-time Audio on Embedded Devices

Elc2011_lorriaux.pdf

Day 2, 2:30am

Magnus Damm

Runtime PM: Upstream I/O Device Power Management

Elc2011_damm.pdf

Jesse Barker

Linux Graphics Meets the ARM Ecosystem

Elc2011_barker.pdf

David Anders (Texas Instruments)

Board Bringup: Open Source Hardware and Software Tools

Elc2011_anders.pdf

Open Tools References

Day 2, 3:40pm

John Williams (PetaLogix)

Dynamic Co-simulation of FPGA-based Linux Systems-on-Chip

Elc2011_williams.pdf

Sumit Semwal (Texas Instruments)

Media Controller Framework (MCF) For OMAP2+ Display Subsystem

Elc2011_semwal.pdf

Day 3, 9:00am

John Stultz (IBM)

Android for Servers?

Elc2011_stultz.pdf

Anand Gadiyar (Texas Instruments)

Tools and Techniques for Debugging Embedded Systems

Elc2011_gadiyar.pdf

Hans Verkuil (Cisco)

Video4linux: Progress, New videobuf2 Framework and the Future

Elc2011_verkul.odp

Day 3, 10:00am

Yoshiya Hirase (Nokia)

Faster Resume For More Energy Savings on MeeGo

Elc2011_hirase.pdf

Jake Edge (Linux Weekly News)

What Embedded Linux Developers Should Know About IPv6

Elc2011_edge.pdf

Grégoire Gentil (Always Innovating)

Hot Multi-OS Switch: How to run Ubuntu, ChromiumOS, Android at the Same Time on an Embedded Device

ELC-AlwaysInnovating-Gentil.pdf

Day 3, 11:00am

Xi Wang (Broadcom)

Controlling Memory Footprint at All Layers: Linux Kernel, Applications, Libraries and Toolchain

Elc2011_xi_mem.pdf

Tom Zanussi and Saul Wold

Building Custom Embedded Images with Yocto

Elc2011_zanussi_wold.odp

Day 3, 2:30pm

Philip Balister

A High Performance Interface Between the OMAP3 and an FPGA

Omap3-fpga.pdf

Jean Pihet (NewOldBits.com)

The Evolution of Tracing and Profiling for Power Management and Accelerators

Elc2011_pihet.pdf

Day 3, 3:40pm

Elizabeth Flanagan (Intel)

Delivering Predictability: The Yocto Project Autobuilder, Automated Sanity Testing, License Collection, and Build Statistics Tracking

Elc2011_flanagan.pdf

Mythri pk

Bringing up HDMI Display for OMAP4 Panda Board - Design, Challenges and Lessons Learned

HDMI_ELC_mythripk.pdf

Day 3, 4:40pm

Khem Raj

Debug/Develop uClibc with QEMU

Elc2011_raj_qemu.pdf

Guntur Ravi Sankar (Samsung)

What are and How to find a program's unused DSOs

ELC_2011_Ravi.pdf

Birds of a Feather Session

Birds-of-a-Feather Sessions

Presenter(s)	Session Description	Presentation
Bill Traynor	elinux.org wiki BOF	Elinux_org_BOF.pdf
Jeff Osier-Mixon (Intel)	Yocto Project Community BoFs (Notes)	No slides.
Luca Coelho (Texas Instruments)	OpenLink WLAN Hacking Workshop	Openlink Workshop slides (PDF file)
Alison Chaiken (Nokia)	"MeeGo BoFs: New Linux Platform for Mobile Computing Devices" (ODP format)	MeeGo BoF Slides (PDF file)
Jason Kridner (Texas Instruments)	BeagleBoard Hands-On Workshop	
Yoshitake Kobayashi (Toshiba)	Moving Forward: Overcoming Compatibility Issues BoFs	Elc2011_kobayashi.pdf

Instructions for Presenters

Please create a link in the table for your presentation, copying the style of other links. (You may need to create an account in order to edit the wiki or upload files.)

When you have created the link, click on it to upload the file containing your slides.

Categories:

- ELC
- 2011
- Events

- Presentations

> From: eLinux.org

ELC 2013 Presentations

Presenters, Demo-ers, Participants: Thanks very much for your participation in Linux Foundation's Embedded Linux Conference 2013.

This page is for collecting the presentations that were made at the conference. During and after the conference we will collect materials from the presenters and place them here. Please watch this page if you are interested in a particular presentation - and if it doesn't show up, please send me and email and we'll try to track it down.

Contents

Videos

Many videos for ELC2013 are available at: http://video.linux.com/categories/2013-embedded-linux-conference-1

In addition, Free Electrons has also provided video of the ELC talks: ELC

Instructions

Presenters: Please post your technical conference presentations on this page. (See Instructions below the tables)

Table of Presentations

NOTE: If you add a wikilink to your presentation and attempt to upload it via the link, it may fail. If it does, use the Special:Upload page to upload your file.

Keynotes

Keynotes

Presenter(s)	Session Description	Presentation	Transcript Status
Jim Zemlin, Executive Director, The Linux Foundation George Grey, CEO, Linaro	Working Together to Accelerate Linux Development		
Andrew Chatham, Google	Google's Self-Driving Cars: The Technology, Capabilities & Challenges		
Dave Stewart, Intel	Code Sweat: Embed with Nightmares		
SpaceX - Moore's Law to Mars	Robert Rose, SpaceX		

Presenters

Presentations

Session Description

Presenter(s)

Presentation

Transcript Status

Day 1, 11:00am

Anatomy of the arm-soc git tree

Olof Johansson, Google

PDF

Beaglebone: The Perfect Telemetry Platform?

Matt Ranostay, Ranostay Industries

PDF

Using and Understanding the Real-Time Cyclictest Benchmark

Frank Rowand, Sony Network Entertainment

PDF

Day 1, 12:00pm

Anatomy of an Embedded KMS Driver

Laurent Pinchart, Ideas on board SPRL

PDF

Kernel Dynamic Memory Allocation Tracking and Reduction

Ezequiel Alfredo Garcia, VanguardiaSur

PDF

The OpenEmbedded Project 2 Years After Adopting the Yocto Project

Koen Kooi, CurcuitCo Electronics

PDF

Day 1, 2:00pm

How to Cook the LTSI Kernel with Yocto Recipe

Hisao Munakata, Renesas Electronics

PDF

PinControl and GPIO Update

Linus Walleij, ST-Ericsson

PDF

Understanding PREEMPT_RT (The Real-Time Patch

Steven Rostedt, RedHat

PDF

Beaglebone Hands-On Tutorial Sessions 1 Sponsored by BeagleBoard.org and BeagleBoardToys.com

Jayneil Dalal, Texas Instruments

PDF

Day 1, 3:00pm

Building a Custom Linux Distribution with the Yocto Project

Sean Hudson, Mentor Graphics

PDF

Common Clock Framework: How to Use It

Gregory Clement, Free Electrons

PDF

RFC: Obtaining Management Buy-in for Mainline Development

Kevin Chalmers, Texas Instruments

PDF

BeagleBone Hands-on Tutorial Session 2 Sponsored by BeagleBoard.org and BeagleBoardToys.com

Jayneil Dalal, Texas Instruments

PDF

Day 1, 4:00pm

Atom for Embedded Linux Hackers and the DIY Community

Scott Garman, Intel Open Source Technology Center

PDF

Controlling Multi-Core Race Conditions on Linux/Android

Mike Anderson, The PTR Group, Inc.

PDF

Making Linux do Hard Real-Time

Brent Roman, Monterey Bay Aquarium Research Institute

PDF

BeagleBone Hands-On Tutorial Session 3 Sponsored by BeagleBoard.org and BeagleBoardToys.com

Jayneil Dalal, Texas Instruments

PDF

Day 1, 5:00pm

How to Decide the Linux Kernel Version for the Embedded Products to Keep Maintaining Long Term

Tzugikazu SHibata, NEC

PDF

Optimizing GStreamer Video Plugins: A Case Study with Renesas SoC Platform

Katsuya Matsubara, IGEL Co., Ltd.

PDF

Your New ARM SoC Linux Support Checklist!

Thomas Petazzoni, Free Electrons

PDF

Day 2, 10:30am

Kernel Testing Tools and Techniques

Matt Porter, Texas Instruments, Inc.

PDF

Debugging on a Production System

Tristan Lelong, Adeneo Embedded

PDF

FIT Image Format Inspired by the Kernel Device Tree Interface

Joel Fernandes, Texas Instruments, Inc.

PDF

Pre-built Binary Toolchains in the Yocto Project

Denys Dmytriyenko, Texas Instruments, Inc.

PDF

Day 2, 11:30am

Extending the swsusp Hibernation Framework to ARM

Russell Dill, Texas Instruments, Inc.

[PDF]

LLVMLinux: Compiling the Linux Kernel with LLVM

Behan Webster, Converse in Code, Inc.

PDF

Making Gadgets Really "cool"

Noor Ul Mubeen, Intel Technology India Pvt Ltd

PDF

Survey of Linux Kernel Debugging Techniques

Kevin Dankwardt, K Computing

PDF

Day 2, 1:45pm

Application Diversity Demands Accelerated Linux Innovation

Mark Orvek, Linaro

PDF

Can You Market an Open Source Project?

Tracey Erway, Intel Corporation Nithya Ruff, Synopsys

PDF

The End of Embedded Linux (as we know it)

Chris Simmonds, 2net Limited

PDF

Toybox: Writing a new Linux Command Line from Scratch

Rob Landley, Multicellular

presentation outline (txt)

Day 2, 2:45pm

Deadline Miss Detection with SCHED_DEADLINE

Yoshitake Kobayashi, TOSHIBA Corporation

PDF, Source code

Embedded Linux Takes on the Hard Problems of Automotive

Alison Chaiken, Mentor Embedded Software Division

PDF

Open Graphics with the Yocto Project

Ross Burton, Intel

PDF

Using GStreamer for Seamless Off-Loading Audio Processing to a DSP

Ruud Derwig, Synopsys

PDF

Day 2, 4:00pm

Designing for Optimisation

Mans Rullgard, ARM/Linaro

PDF

Namespaces for Security

Jake Edge, LWN.net

PDF

Yocto Project Overview and Update

Saul Wold, Intel

PDF

Day 2, 5:00pm

Board Bringup: You, Me, and I2C

David Anders, Texas Instruments

PDF - Resource Page

System-wide Memory Management without Swap

Howard Cochran, Lexmark International

PDF

Target Communication Framework: One Link to Rule Them All

Anna Dushistova, Me, Myself, and I

PDF

Day 3, 9:00am

EasyUI: No Nonsense Mobile Application Development with EFL

Leandro Pereira, ProFUSION Embedded System

PDF

In Kernel Switcher: A Solution to Support ARM's New big.LITTLE implementation

Mathieu Poirier, Linaro

PDF

Yocto Meta-Virtualization Layer Project

Michael Christofferson, Enea

PDF

Embedded Android Workshop

Karim Yaghmour, Opersys

PDF

Day 3, 10:00am

F2FS (Flash-Friendly File System)

Joo-Young Hwang, Samsung Electronics Co., Ltd.

PDF

Lessons Learned in Designing a Self-Video, Self-Hovering Nano Copter

Gregoire Gentil, Always Innovating

PDF

Leveraging Linux - Code Coverage for Post-Silicon Validation

Mehdi K., UBC Integrated Systems Design Lab

PDF

Day 3,11:15am

Bringing kconfig to EGLIBC

Khem Raj, OpenEmbedded

PDF

Gentoo-Bionic: We Can Rebuild Him. Better. Stronger. Faster.

Christopher Friedt, Research In Motion

PDF

Task Scheduling for Multicore Embedded Devices

Gap-Joo Na, Electronics and Telecommunications Research Institute (ETRI)

PDF

Day 3, 12:15pm

Adventures in (simulated) Assymmetric Scheduling

Pantelis Antoniou, Antoniou Consulting

PDF

An Insight into the Advanced XIP Filesystem (AXFS)

Aditya Kumar, Sony India Software Centre Pvt Ltd.

PDF

Tips of Malloc and Free

Tetsuyuki Kobayashi, Kiyoto Microcomputer

PDF

Day 3, 2:15pm

How to Build Your Own Digital Signage Solution with Yocto Project

Nitin Kamble, Intel Corporation

PDF

Leveraging SPDX with Yocto

Mark Gisi, Wind River Systems Mark Hatle, Wind River Systems

PDF

The 'Embedded' Problem as Experienced by Intel's Reference Phones

Mark Gross, Intel/MCG/PSI

PDF

Day 3, 3:15pm

olibc: Another C Runtime Library for Embedded Linux

Jim Huang, 0xlab

PDF

Security Best Practices for Embedded Systems

John Mehaffey, MontaVista Software

PDF

uCLinux for Custom Mobile Devices

Hunyue Yau, HY Research LLC

PDF

Workshops

Workshops

Presenter(s)	Session Description	Presentation
Filip Thoen, Synopsys	Getting Linux BSPs Developed Faster	
Yocto Project Developers	Tocto Project & OpenEmbedded BoF	
Matt Porter, Texas Instruments	Kernel Testing Tools and Techniques	
Jesse Barker, Linaro	Common Display Framework	

Instructions for Presenters

Please create a link in the table for your presentation, copying the style of other links. (You may need to create an account in order to edit the wiki or upload files.)

When you have created the link, click on it to upload the file containing your slides.

Categories:

- ELC
- 2013
- Events
- Presentations

From: eLinux.org

ELC 2014 Presentations

Here are the presentations from ELC 2014.

Some videos are available, courtesy of

- the fine folks at Free Electrons
- Frank Rowand and Tim Bird of Sony Mobile Communications

The "Free Electrons Videos" links in the table below are to their hosting page:

http://free-electrons.com/blog/elc2014-videos/

The "YouTube" links in the table below are directly to the respective video. The ELC 2014 YouTube playlist is at:

https://www.youtube.com/playlist?list=PL_xRWvMmKDbLAOMqMDovw-4F9fEYmf15s

Contents

Table of Presentations

NOTE: If you add a wikilink to your presentation and attempt to upload it via the link, it may fail. If it does, use the Special:Upload page to upload your file.

Keynotes

Keynotes

Presenter(s)	Session Description	Presentation	Transcript Status	Video
Tim Bird, Sony Mobile	The Paradox of OpenSource and Embedded	Prezi PDF		
Karim Yaghmour, CEO-Opersys David Anders, Senior Embedded Systems Engineer at CircuitCo Tim Bird, Senior Software Engineer at Sony Mobile Matt Porter, Technical Lead at Linaro Benjamin Zores, Software Architect at Alcatel-Lucent	Panel: IoT and the Role of Embedded Linux and Android - David Anders, CircuitCo; Tim Bird, Sony Mobile; Matt Porter, Linaro; Benjamin Zores, Alcatel-Lucent; Karim Yaghmour, OperSys (Moderator)			
Tim Bird, Sony Mobile	Closing Game			YouTube

Presenters

Presentations

Session Description

Presenter(s)

Presentation

Transcript Status

Video

Day 1, 10:30am

A Deep Dive Into DEX File Format

Rodrigo Chiossi, Intel Open Source Technology Center

PDF

ROM Cooking and Good Practices

Jeremy Vagnet, Genymobile

PDF

LTSI Project Update for 3.10 Kernel and Future Plans

Hisao Munakata, Renesas

PDF

microYocto and the 'Internet of Tiny'

Tom Zanussi, Intel Open Source Technology Center

PDF

YouTube

USB and the Real World

Alan Ott, Signal 11 Software

PDF

Free Electrons Videos

Day 1, 11:30am

Update on Boot Time Reduction Techniques with Figures

Michael Opdenacker, Free Electrons

PDF

Free Electrons Videos

Finding Performance and Power Issues on Android Systems

Eric Moore, Intel Corporation

PDF

YouTube

Running Code in the Android Stack

Karim Yaghmour, Opersys

Mastering the DMA and IOMMU APIs

Laurent Pinchart, Renesas Linux Kernel Team

PDF

YouTube

Qt5 and Yocto - Adding SDK and Easy App Migration for Qt4

Dmytriyenko, Texas Instruments

PDF

Day 1, 2:00pm

The Android Graphics Path, In Depth

Chris Simmonds, 2net

PDF

The Android Media Framework - A Deep Dive

Poornachandra Kallare, TPVision

PDF

Device Tree for Dummies

Thomas Petazzoni, Free Electrons

PDF

YouTube

Porting Linux to a New Architecture

Marta Rybczynska. Kalray

PDF

YouTube

User Space Drivers in Linux - Pros, Cons, and Implementation Issues

Michael Christofferson, Enea

Day 1, 3:00pm

Android without Java

Bernard Rosenkranzer, Linaro

PDF

Improving Performance of Key External Projects Used in Android

Khasim Syed Mohammed, Linaro

PDF

How to Build a Linux-based Robot

Michael E. Anderson, The PTR Group, Inc.

PDF

YouTube

Using Real-Time Patch with LTSI Kernel

Yoshitake Kobayashi, Toshiba

PDF

YouTube

Yocto Project/OpenEmbedded BoF

Jeff Osier-Mixon, Intel Corporation

Day 1, 4:20pm

Android without Java (Cont.)

Bernard Rosenkranzer, Linaro

PDF

X86 ROM Cooking 101: A Hands on Tutorial

Ron Munitz, Nubo

Cooking PDF ; Build System PDF

A Remote Power Analyzer Based on Power Modeling in Heterogenous Embedded Linux Systems

Young-Joo Kim, Electronics and Telecommunications Research Institute

Making a Splash: DIgital Signage Powered by Minnowboard and the Yocto Project

Nitin Kamble and John Hawley, Intel Open Source Technology Center

PDF

Supporting a New ARM Platform: The Allwinner SoCs Example

Maxime Ripard, Free Electrons

PDF

Free Electrons Videos

Day 1, 5:20pm

Headless Android Strikes Back

Gary Bisson, Adeneo Embedded

PDF

Engaging Device Trees

Geert Uytterhoeven, Glider bvba

PDF

YouTube

Fear and Loathing in the Media Transfer Protocol

Linus Walleij, Linaro

PDF

Free Electrons Videos

Use-Case Power Management Optimization: Identifying and Tracking Key Power Indicators

Patrick Titiano, BayLibre

PDF

YouTube

Day 1, 6:30pm

(BoFs) Device Tree: Best Practices, Review Process & Maintainership Discussions

Olof Johansson, Google

YouTube

Day 2, 10:30am

Android Platform Debugging and Development

Karim Yaghmour, Opersys

Tuning Android for Low RAM

Chris Simmonds, 2net

PDF

(Tutorial) Some GCC Optimizations for Embedded Software

Khem Raj, Juniper Networks

PDF

Linux for Microcontrollers: Spreading the Disease

Vitaly Wool, Softprise Consulting OU

PDF

Trees need care: A Solution to Device Tree Validation Problem

Tomasz Figa, Samsun R&D Institute

PDF

YouTube Free Electrons Videos

Day 2, 11:30am

Android KitKat Internals

Benjamin Zores, Alcatel-Lucent

PDF

Security Enhancements (SE) for Android

Stephen Smalley, US National Security Agency

PDF

(Tutorial) Some GCC Optimizations for Embedded Software (Cont.)

Khem Raj, Juniper Networks

PDF

Can Board Bringup Be Less Painful with Yocto and Linux?

Insop Song, Gainspeed

PDF

Extending Linux using Arduinos

Michael E Anderson, The PTR Group, Inc.

PDF

Free Electrons Videos

(BoFs) QCOM SoC Mainlining

Tim Bird, Sony Mobile

YouTube

Day 2, 2:00pm

Android App Tuning Techniques Workshop

Mark Murphy, CommonsWare

Making Android More Wearable: The Challenges of Adding Multi-SPorts Sensors and Radios

Gil Zhaiek, Recon Instruments

PDF

A Timeline for Embedded Linux

Chris Simmonds, 2net

PDF

YouTube

An Introduction to the Video4Linux Framework

Hans Verkuil, Cisco Systems Norway

PDF

Introducing Embedded Linux to Universities

Victor Rodriguez, Intel

PDF

Day 2, 3:00pm

Android App Tuning Techniques (Cont.)

Mark Murphy, CommonsWare

Android Middleware Development

Euler Rachid, Samsung R&D Brazil Lab

Debugging - Linux Kernel Testing

Matt Porter, Linaro

Free Electrons Videos

Hardware-Assisted Software Tracing

Adrien Verge, Ecole Polytechnique Montreal

PDF

YouTube

LLVMLinux: Embracing the Dragon

Behan Webster, Converse in Code Inc.

PDF

Day 2, 4:20pm

Android App Tuning Techniques Workshop (cont.)

Mark Murphy, Commonsware

Android on Atom for Teeth Health

Rafael Coutinho, PhiInnovations]

PDF

Debugging (Sigrok: Using Logic to Debug Logic)

Matt Ranostay, Intel Open Source Technology Center David Anders, CircuitCo

PDF

Free Electrons Videos

Two Years of ARM SoC Support mainlining: Lessons Learned

Thomas Petazzoni, Free Electrons

PDF

Free Electrons Videos

webOS, An OpenEmbedded Use Case

Iyad Qumei, LG Electronics

PDF

(BoFs) ARM/ARM64: Code Sharing, Consolidation & Upstreaming

Olof Johansson, Google

YouTube

Day 2, 5:20pm

Android App Tuning Techniques Workshop (Cont.)

Mark Murphy, Commonsware

Bringing the BlueZ Back to Android

Marcel Holtmann, Intel's Open Source Technology Center

PDF

Building Tools From the Outside In: Bringing User-Centered Design to Embedded Linux

Belen Barros Pena, Intel's Open Source Technology Center

PDF

YouTube

Debugging - Panel Discussion

Karim Yaghmour, Opersys Matt Ranostay, Intel David Anders, CircuitCo Matt Porter, Linaro

Free Electrons Videos

System Power Management Interface (SPMI)

Josh Cartwright, Qualcomm Innovation Center

YouTube

Day 3, 9:00am

Embedded Android Workshop

Karim Yaghmour, Opersys

The Growth of Android in Embedded Systems

Benjamin Zores, Alcatel-Lucent

PDF

Software True Random Number Generator for Embedded Linux

Nicholas McGuire, Safety Critical Linux Working Group of OSADL

Understanding the Embedded Linux Ecosystem with Codeface

Wolfgang Mauerer, Siemens

PDF

YouTube

Volatile Ranges

Mincham Kim, LG Electronics

PDF

Day 3, 10:00am

Embedded Android Workshop (Cont.)

Karim Yaghmour, Opersys

Extending Android via Extenal Microprocessors

Mike Anderson, The PTR Group

PDF

YouTube

Linux Quickboot

Tristan Lelong, Adeneo Embedded

PDF

Free Electrons Videos

Open Source Tools for Software Defined Radio on Multicore ARM+DSP

Philip Balister, OpenSDR

PDF

YouTube

Ubuntu Touch Internals

Ricardo Salveti de Araujo, Ubuntu Touch LowLevel Stack

PDF

YouTube

Day 3,11:25am

Embedded Android Workshop (Cont.)

Karim Yaghmour, Opersys

Here There Be Dragons: Using cland/LLVM to Build Android

Behan Webster, Converse in Code

PDF

Hardware Accelerated Video Streaming with V4L2

Gabriel Huau, Adeneo Embedded

PDF

SMP Bring Up On ARM SOCs

Gregory Clement, Free Electrons

PDF

Free Electrons Videos

The #qt/#wayland/#systemd/#btrfs-phone....the Jolla Phone

David Greaves, Mer Project

PDF

Day 3, 2:00pm

Multiwindow Support on Android

Andrzej Wieczorek, Tieto Mikel Echegoyen, Tieto

PDF

Using Chroot to Bring Linux Applications to Android

Mike Anderson, PTR Group

PDF

Buildroot: What's New?

Thomas Petazzoni, Free Electrons

PDF

Free Electrons Videos

Improving Performance of a WebKit Port MIPS Platform

Adrian Perez de Castro, Igalia

PDF

YouTube

Using Yocto for Modules Manufacturers

Alexandre Belloni, Free Electrons

PDF

Free Electrons Videos

Day 3, 3:00pm

Multi Persona Android

Amir Goldstein, Cellrox

PDF

Collaborative GPL Enforcement Through Non-Profit Entities

Bradley M. Kuhn, Software Freedom Conservancy

PDF

Productizing Telephony and Audio in a GNU/Linux (Sailfish OS) Smartphone

Martti Piirainen, Tieto

PDF

Free Electrons Videos

What's Going on with SPI

Mark Brown, Linaro

PDF

YouTube

Day 3, 3:50pm

Genivi and AGL - A View From the Side

Vitaly Bordyug, Mentor Graphics

Kernel USB Gadget Configfs Interface

Matt Porter, Linaro

PDF

Free Electrons Videos

Using Agile Development Practices for Kernel Development

Chase Maupin, Texas Instruments

PDF

YouTube

Instructions for Presenters

Please create a link in the table for your presentation, copying the style of other links. (You may need to create an account in order to edit the wiki or upload files.)

When you have created the link, click on it to upload the file containing your slides.

Categories:

- ELC
- 2014
- Events
- Presentations

> From: eLinux.org

ELC 2015 Presentations

Presentations from ELC 2015.

Contents

Videos

The "YouTube" links in the table below are directly to the respective video. The ELC 2015 YouTube playlist is at:

http://www.youtube.com/playlist?list=PLGeM09tlguZTPUxEvsQiDgX0XDjfOL6oR

Table of Presentations

NOTE: If you add a wikilink to your presentation and attempt to upload it via the link, it may fail. If it does, use the Special:Upload page to upload your file.

Presenters

Day 1 Presentations

Session Description

Presenter(s)

Presentation

Transcript Status

Video

Day 1, 9:00am

Driving standards and Open Source to Grow the Internet of Things

Mark Skarpness, Director of Systems Engineering at Intel

PDF

YouTube

Day 1, 9:30am

Project Ara

Paul Eremenko, Head of Project Ara, ATAP at Google & Marti Bolivar, Project Ara Software Lead, Google

PDF

YouTube

Day 1, 10:30am

Android OTA Updates

Andrew Boie, Intel

PDF

YouTube

Generalizing Android for Low-Cost 64-Bit ARM-Based Community Boards

Khasim Syed Mohammed, Linaro

PDF

YouTube

An Overview of the Kernel DMAEngine Subsystem

Maxime Ripard, Free Electrons

PDF

YouTube

Tuning systemd for Embedded

Alison Chaiken, Mentor Graphics

PDF

YouTube

The Open Interconnect Consortium (OIC) Security Model and Vision

Ned Smith, Intel

PDF

YouTube

Day 1, 11:30am

Build and Distributing SDK Add-Ons

Dave Smith, NewCircle

PDF

YouTube

Fuzzing the Media Framework in Android

Alexandru Blanda, Intel

PDF

YouTube

Autonomous Navigation for an OMAP4 Nano-Drone

Grégoire Gentil, Always Innovating

PDF

YouTube

Buildroot: Embedded Linux for Small Devices and Makefile Enthusiasts

Stephanie Lockwood-Childs, VCT Labs

SVG

YouTube

LTSI: Latest Status and Kernel Testing

Tsugikazu Shibata, NEC

PDF

YouTube

Virtualization for Small Devices

Jesse Zbikowski and Stephan Okay, Cratus technologies

PDF

Day 1, 2:00pm

Solving Global Illiteracy With Android and XPRIZE

Jono Bacon, XPRIZE

PDF

YouTube

Anatomy of a Screenshot

Rodrigo Chiossi, Intel

PDF

YouTube

Using Intel Edison to Fuse Embedded Linux With Existing Drone Flight Controllers

Mark F. Brown, Intel & Joel Rosenzweig, Intel

PDF

YouTube

Bluetooth 4.2 - New Features for Linux and IoT

Marcel Holtmann, Intel

PDF

YouTube

The Device Tree as a Stable ABI: A Fairy Tale?

Thomas Petazzoni, Free Electrons

PDF

YouTube

IoTivity and Embedded Linux Support

Kishen Maloor, Intel

PDF

YouTube

Day 1, 3:00pm

Android's New Stream-Based Camera Architecture

Balwinder Kaur, ON Semiconductor

PDF

YouTube

Chaining HALs

Hunyue Yau, HY Research

PDF

YouTube

Performance Analysis Using the perf Suite

Mans Rullgard

PDF

YouTube

Optimize uClinux for ARM Cortex-M4

Jim Huang, South Star Xelerator & Jeff Liaw, National Cheng Kung University

PDF

YouTube

10 Years of Open Source Robotics

Laurent Pinchart, Ideas on Board

PDF

YouTube

The OpenDOF Project - An Open Distributed Object Framework For The IoT

Bryant Eastham, Panasonic

PDF

YouTube

Day 1, 4:20pm

Implementation of the Global Task Scheduler in big.LITTLE Android Platforms

Michael E. Anderson, The PTR Group

PDF

YouTube

Utilizing the Android Open Source Project to Support Controllers for Single-Use Devices. (X-Ray Guns! Pew Pew!)

Ben Friedberg, SDG Systems

PDF

YouTube

Flying Penguins: Embedded Linux Applications for Autonomous UAVs

Clay McClure

PDF

YouTube

Introduction to IEEE 1588 Precision Time Protocol (PTP) Using Embedded Linux Systems

Insop Song, Gainspeed

PDF

YouTube

Poky meets Debian: Understanding How to Make an Embedded Linux by Using an Existing Distribution's Source Code

Yoshitake Kobayashi, Toshiba

PDF

YouTube

Ready-Made Recipes to Add Security and Data

Dominig ar Foll, Intel

PDF

YouTube

Day 1, 5:20pm

Memory Management Internals

Karim Yaghmour, Opersys

PDF

YouTube

Android Multilib Build Cheat Sheet

Amit Pundir, Linaro

PDF

YouTube

Teaching More Fish to Fly

John Hawley, Intel

PDF

YouTube

Automated Flashing and Testing for Continuous Integration

Igor Stoppa, Intel

PDF

YouTube

Tizen: System-Wide Memory Defragmenter Without Killing Any Application

Pintu Kumar, Samsung

PDF

YouTube

Getting Started with AllJoyn

Ivan R. Judson, Microsoft

PDF

YouTube

Day 2 Presentations

Session Description

Presenter(s)

Presentation

Transcript Status

Video

Day 2, 9:00am

Android Verified Boot

Andrew Boie, Intel

PDF

YouTube

Heterogeneous Multi-Core Architecture Support for Dronecode

Mark Charlebois, Qualcomm Innovation Center (QuIC)

PDF

YouTube

Understanding Embedded Linux Benchmarking Using Kernel Trace Analysis

Alexis Martin, Inria

PDF

YouTube

A Scalable, Cloud-based Device Reprogramming Architecture

James Simister, Panasonic

PDF

YouTube

Customizing AOSP for my Device

Rafael Coutinho, Phi Innovations

PDF

YouTube

Building Multi-Processor FPGA Subsystems – Allowing Linux to Supervise Embedded Real-Time Processing Systems

Chris Martin, Altera

PDF

YouTube

Day 2, 10:00am

Implementing Controls with Bluetooth SMART in Android

Michael E. Anderson, The PTR Group

PDF

YouTube

Open Source Drones on Linux

Lorenz Meier MLC/TLC

PDF

YouTube

NAND Support: (New?) Challenges for the MTD/NAND Subsystem

Boris Brezillon, Free Electrons

PDF

YouTube

Building IoT systems with openHAB

Matt Porter, Konsulko

PDF

YouTube

Day 2, 11:20am

Room For Cooperation: Bionic and musl

Bernhard Rosenkränzer, Linaro

PDF

YouTube

Aster: A Remote GUI Control Tool for the Android Platform

Yongqin Liu, Linaro

PDF

YouTube

Application of Data Fusion to Aerial Robotics

Paul Riseborough, 3DRobotics

PDF

YouTube

Embedded Distributed Systems: A Case of Study

Victor Rodriguez, Intel

PDF

YouTube

Transitioning From uclibc to musl for Embedded Development

Rich Felker, Openwall

PDF

YouTube

Security Architecture in the IOT Age

Stephen Arnold, VCT Labs

PDF

YouTube

Day 2, 1:40pm

Dronecode Project and Autopilot With Linux

Andrew Tridgell, Technical Steering Committee Chair of Dronecode Project

http://uav.tridgell.net/ELC_2015/

YouTube

Day 2, 2:10pm

IoT Panel

Dominig Ar Foll, Intel (Tizen); Greg Burns, AllSeen Alliance; Bryant Eastham, Panasonic; Guy Martin, Samsung; Tim Bird, Sony Mobile (Moderator)

YouTube

Day 2, 3:25pm

Platform-Level UI Customization

Karim Yaghmour, Opersys

PDF

YouTube

Upstreaming, Downstreaming, 'Sidestreaming': How Can Android-Based Projects Work Together?

Bernhard Rosenkränzer, Linaro

PDF

YouTube

The Syria Airlift Project: Open-Sourcing Humanitarian Airlift

Mark Jacobsen, U.S. Air Force

PDF

YouTube

Last One Out, Turn Off The Lights

Geert Uytterhoeven, Glider bvba

PDF

YouTube

Generic PHY Framework

Kishon Vijay Abraham, Texas Instruments

PDF

YouTube

Linux for Microcontrollers: From Marginal to Mainstream

Vitaly Wool, Softprise Consulting OU

PDF

YouTube

Day 2, 4:25pm

Android Customization: From the Kernel to the Apps

Cédric Cabessa, Genymobile

PDF

YouTube

Embedded Linux moves into High School Robotics

Michael E. Anderson, The PTR Group

PDF

YouTube

Freedreno Status Report: Upstream and FOSS Graphics on ARM/SoC Devices

Rob Clark, Red Hat

PDF

YouTube

Creating Open Hardware Tools

David Anders, Intel

PDF

YouTube

Sigrok: Adventures in Integrating a Power-Measurement Device

Bartosz Golaszewski, BayLibre

PDF

Building a General Purpose Android Workstation

Ron Munitz

PDF

YouTube

Day 2, 5:25pm

Creating Platform Development Tools

François-Denis Gonthier, Opersys

PDF

YouTube

DroneAPI: A Tutorial on Drone Control

Kevin Hester, 3DRobotics

PDF

YouTube

Regulators: Learning To Play With Others

Mark Brown, Linaro

PDF

YouTube

Open Lighting Architecture: Blinky Lights!

Matt Ranostay, Intel

PDF

YouTube

Real Time Linux Scheduling Performance Comparison

Vince Bridgers, Altera

PDF

YouTube

Day 2, 6:40pm

BoFs: Applying Linux to the Social Infrastructure Systems

Noriaki Fukuyasu, Linux Foundation

PDF

BoFs: Cryptography and Randomness

Jesse Zbikowski

PDF

BoFs: Dronecode Project

Andrew Tridgell & Lorenz Meier

PDF

BoFs: Kernel Wish List

John Stultz

PDF

BoFs: MinnowBoard - John Hawley, Intel BoFs: Yocto Project / OpenEmbedded

Jeff Osier-Mixon

PDF

BoFs: Kernel Testing for Upstream with kernelci.org

Kevin Hilman, Linaro

PDF

Day 3 Presentations

Session Description

Presenter(s)

Presentation

Transcript Status

Video

Day 3, 9:00am

Android Based Penetration Testing Framework

Ron Munitz

PDF

Filesystem Considerations for Embedded Devices

Tristan Lelong, Adeneo

PDF

YouTube

Embedded Digital TV Kernel Pipelines via Media Controller API

Mauro Carvalho Chehab,

PDF

YouTube

Introduction to Kernel Power Management

Kevin Hilman, Linaro

SVG

YouTube

Transactional Device Tree & Overlays: Making Reconfigurable Hardware Work

Pantelis Antoniou, Konsulko Group

PDF

YouTube

Embedded Android Workshop

Karim Yaghmour, Opersys

PDF

YouTube

Day 3, 10:00am

Maintaining Multiple Android Linux Kernels at Intel

Mark Gross, Intel

PDF

YouTube

Creating Eco-System for R-Car LCB

Hisao Munakata, Renesas

PDF

YouTube

Maintaining a Large Kernel Subsystem

Arnd Bergmann

PDF

YouTube

Status Report for IEEE 802.15.4 and 6LoWPAN in Linux

Stefan Schmidt, Samsung

PDF

YouTube

What's New with Toybox

Rob Landley

TXT

YouTube

Day 3, 11:20am

Android and Modern Toolchains: gcc 5.0, clang 3.6 and binutils 2.25

Bernhard Rosenkränzer, Linaro

PDF

Embedded Android Workshop (Cont.)

Karim Yaghmour, Opersys

PDF

YouTube

Linux in the Connected Car Platform

Daniel Jackson, Dialexa

PDF

YouTube

Getting Started with Embedded Linux: Using the Yocto Project to Build your Own Custom Embedded Linux Distribution

Rudolf (Rudi) Streif

PDF

YouTube

Linux Kernel Selftest Framework - Quality Control for New Releases

Shuah Khan, Samsung

PDF

YouTube

Overcoming Obstacles to Contributing to Linux

Tim Bird, Sony Mobile

PDF

YouTube

Day 3, 1:40pm

Doing big.LITTLE right: little and Big Obstacles

Vitaly Wool & Vlad Rezki, Softprise Consulting OU

PDF

YouTube

Getting Started with Embedded Linux: Using the Yocto Project to Build your Own Custom Embedded Linux Distribution (Cont.)

Rudolf (Rudi) Streif

PDF

Improving the Embedded Linux Development Workflow

Paul Eggleton, Intel

PDF

YouTube

Testing Video4Linux Applications and Drivers

Hans Verkuil

PDF

YouTube

Embedded Android Workshop (Cont.)

Karim Yaghmour, Opersys

PDF

YouTube

Shrinking C Code

Rob Landley

Rob said to point people here for content: TXT

YouTube

Day 3, 2:40pm

Fixing the y2038 Bug

Arnd Bergmann, Linaro

PDF

Enhancing Real-Time Capabilities with the PRU

Rob Birkett, Texas Instruments

PDF

YouTube

Fastboot Tools and Techniques

John Mehaffey, Mentor Graphics

PDF

YouTube

The Ephemeral Smoking Gun: Using ftrace and kgdb to Resolve a pthread 'deadlock'

Brad Mouring, National Instruments

PDF

YouTube

Day 3, 4:00pm

Embedding Openness in the Connected Car

Matt Jones, Senior Infotainment Specialist at Jaguar Land Rover

PDF

YouTube

Day 3, 4:30pm

Community Involvement: Looking Forward and Looking Back

Deepak Saxena, Noted Linux Kernel Developer

PDF

Technical Showcase Posters

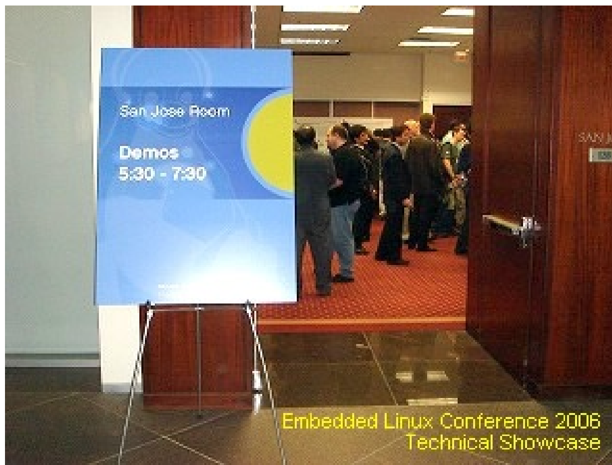
Poster Title	Presenter	Poster
96Boards HiKey	Linaro	PDF
Embedded Linux build system - Buildroot	Thomas Petazzoni	PDF
FIRST Robotics Competition	FRC Team 2643 - Dark Matter	PDF
FIRST Robotics Linux-based Controller	Mike Anderson / The PTR Group, Inc.	PDF
Fluent Bit: Data Collector for Embedded Linux	Eduardo Silva, Treasure Data	PDF
Freedreno - Open Source ARM Graphics	Rob Clark	PDF
OpenDOF Project "One Page" IoT	Bryant Eastham, Panasonic	PDF
Power measurement with sigrok & ACME	Bartosz Golaszewski, Patrick Titiano / BayLibre, Ingenios	PDF
Scalable, Cloud-Based Device Reprogramming	James Simister - Panasonic	PDF
Sony Mobile phone debug board	Werner Johansson	PDF
The Syria Airlift Project Open-Sourcing Humanitarian Airlift	Mark Jacobsen, Jessie Mooberry	PDF
Toaster - A web interface to the Yocto Project	Paul Eggleton, David Reyna, Jeffrey Osier-Mixon - Yocto Project	PDF
USRP E310 - Embedded Software Defined Radio	Philip Balister, Moritz Fischer, Jonathon Pendlum	PDF
VR Spark - Drone Code Edition	Roberto Navoni	PDF

Categories:

- 2015
- ELC

> From: eLinux.org

ELCE 2010 Technical Showcase



Contents

Embedded Linux Conference Europe 2010 Technical Showcase

We are happy to be planning a technical showcase again this year at Embedded Linux Conference Europe. This Wiki page will guide you in giving your demonstration. For conference details, please follow this link to the conference website:

- http://embeddedlinuxconference.com/elc_europe10



Details

The demo session will be on Thursday, October 28th, between 12:35 and 14:30 in room Darwin. See [the agenda].

The demo room will be available starting at ***8:00 pm*** for demo setup.

Objective

"Seeing is believing!" Unlike in Desktop or Server applications of Linux, in embedded systems Linux technology is usually unseen. Although it is a challenge, we want to expose Linux software and techniques so that other developers who are interested in this technology can learn, and become potential contributors themselves. We hope that through opening up technical details and projects (in the way of Open Source development), these demonstrations will provide insights that are helpful to developers.

Criteria

- No *purely* commercial demonstrations.
- The Embedded Linux Conference Europe should not be used for a pure sales pitch.
- You can show off a product, if there are features or aspects of the product you are displaying that are open source.
- Demonstrations should be of software that is available under GPL or LGPL compatible licenses.

Coordinator

Michael Opdenacker - Free Electrons

- e-Mail : Michael(a)free-electrons (dot) com
- Please replace (a) to "@" and (dot) to "."

What will be prepared by CE Linux Forum

CE Linux Forum will provide the following **FREE OF CHARGE** :

- A Demonstration desk, which is half of a table If you need special arrangements, please contact the coordinator.



- Power supply TBD: UK and or EU standard
- Assuming the power is less than 750W. If your demonstration will exceed that, please consult the coordinator.
- A large-format printed poster, based on the file you submit.
- Please note that a dedicated (wired) Internet connection will **NOT** be prepared. However, wireless Internet may be available via the hotel wireless network. We cannot, however, guarantee Internet access.
- Logistics Support
- Details will be given to the demonstrators

Dimension of the desk

- Please refer to the attached photo.

Poster

All demonstrators are requested to prepare a poster to explain the technical outline of the demonstration. You can download the template from this page. CELF will print them out on large size paper. The document should be sent to the coordinator by October 11th.

- ELCE2010_PosterForm.odp
- ELCE2010_PosterForm.ppt

Caution: Do not change the paper size. It is intentionally set to a big size.

Some tips

The demo consists of a poster (printed by CELF), with a few sentences talking about a tool, technology or distribution and a table top where you can place anything from:

1. nothing at all, just have a person sitting there to talk to people about what's on the poster, or ...
2. something on a laptop - like a program running or a even a static chart showing some result
3. something running on a board or a device
4. something on an actual CE product

You can also put some flyers there for people to grab, if you'd like.

If you can put charts and graphs on the poster, that's nice but it is NOT required. Neither the poster nor the hardware (if any) needs to be elaborate or require lots of setup. One year I showed up with a camcorder and just talked to people about it's bootup time (which I had worked on).

If you are a presenter, the demo session is a good chance to talk to people one-on-one about your session topic. This gives you more time for interactive Q&A, and a chance to show something hands-on, if that applies to your subject.

How to apply

1. Send an e-mail to the coordinator with the following information:
    i. The contact person, the company name or the project name, the outline of the demonstration.
    ii. The coordinator will acknowledge your application by e-mail.
2. Prepare the poster and send the data to CELF Office and the demonstration coordinator by **October 11th**. It will be printed out by the conference organizers on big size paper.
    i. Prepare the demonstration equipment
    ii. You can bring the equipment yourself, or ship it.
    iii. If you do not have a consignee which will be convenient for you, send them to the following address: *Address not available yet*

FAQ

- **Q**: Is this opportunity limited to CELF members?
- **A**: No. You do not have to be a forum member to have a demo at the conference. However, each demo should be associated with a paid conference attendee.
- **Q**: Is the demonstrator required to make a presentation at the conference?
- **A**: No, this is not required. However, we believe there will be many people who will to be interested in details of the technology, so we hope that you can have a session. Please note that the due date for a session proposal is earlier than the due date for demo sign-up.
- **Q**: Should I assign some dedicated staff to carry on the demonstration?
- **A**: No, this is not necessary. There will be a dedicated time slot for the demos that will be separate from the other technical sessions. So there should be no conflict between the demo session and the other technical sessions.
- **Q**: Is there any fee to demo something in this session?
- **A**: No, it is completely free. However, the session is limited to conference attendees, so please register for the conference if you would not otherwise do so. If you have someone coming who will ONLY run or assist with your demo, and not attend any sessions, then that person may be admitted for free, for just the demo session (and setup and take down). However, each demo should be associated with at least one registered attendee.

Category:

- ELCE

> From: eLinux.org

ELCE 2011 Presentations redirect

(Redirected from ELCE 2011 Presentations)

Presenters, Demo-ers, Participants: Thanks very much for your participation in Linux Foundation's Embedded Linux Conference Europe 2011.

This page is for collecting the presentations that were made at the conference. During and after the conference we will collect materials from the presenters and place them here. Please watch this page if you are interested in a particular presentation - and if it doesn't show up, please send me and email and we'll try to track it down.

Contents

Videos

Once again, through the diligent work of the Free Electrons team, all videos for ELCE2011 can be found at ELCE2011 Videos.

Instructions

Presenters: Please post your technical conference presentations on this page. (See Instructions below the tables)

Table of Presentations

NOTE: If you add a wikilink to your presentation and attempt to upload it via the link, it may fail. If it does, use the Special:Upload page to upload your file.

Keynotes

Keynotes

Presenter(s)	Session Description	Presentation
Jim Zemlin, Executive Director of The Linux Foundation	Imagine a World Without Linux	
Linus Torvalds, Alan Cox, Thomas Gleixner, Paul McKenney	Kernel Developer Panel	No presentation.
Antti Aumo, President of Global Solutions at Ixonos	Re-Defining the Cloud Phone	
Dirk Hohndel, Chief Linux and Open Source Technologist at Intel	Reflection on 20 Years of Linux	
Jonathan Corbet, Editor at LWN	The Kernel Report: 20th Anniversary Edition	
Wim Coekaerts, Senior Vice President, Linux and Vistualization Engineering at Oracle	Engineered Systems With Linux	
Matt Jones, Vice President, GENIVI Alliance & Technical Lead - Next Generation Infotainment, Jaguar Land Rover	Linux for In Car Infotainment	

Presenters

Presentations

Presenter(s)

Session Description

Presentation

Day 1, 10:45am

Zach Pfeffer (Linaro)

Linaro's Android Platform

PDF

Thomas Gleixner (Linutronix)

Another Mile Down the RT Road

Jessica Zhang (Intel)

The Yocto Project Eclipse PlugIn: An Effective IDE Environment for both Embedded Application and System Developers

PDF

Day 1, 11:45am

Satoru Ueda (Sony/Japan OSS Promotion Forum)

Contributing to the Community? Does your manager support you?

Benjamin Zores (Alcatel-Lucent)

Embedded Linux Optimization Techniques: How Not To Be Slow

PDF

Ohad Ben-Cohen (Texas Instruments/Wizery)

Asymmetric Multiprocessing using VirtIO (was: "Remote Processor Messaging")

PDF

Day 1, 2:00pm

Jeff Osier-Mixon (Intel)

Collaborative Initiatives in Embedded Linux

PDF

Karim Yaghmour (Opersys, Inc.)

Leveraging Android's Linux Heritage

Pierre Tardy (Intel)

Using pytimechart For Real World Analysis

PDF

Day 1, 3:00pm

Arnd Bergmann (Linaro)

Optimizations for Cheap Flash Media

Vitaly Wool (Sony Ericsson)

Saving Power With Wi-Fi: How to Prolong Your Battery Life and Still Stay Connected

ODP

David Stewart (Intel)

Developing Embedded Linux Devices Using the Yocto Project and What's New in 1.1

PDF

Day 1, 4:15pm

Tetsuyuki Kobayashi (Kyoto Micro Computer)

Android is NOT Just "Java On Linux"

PDF

Thomas Petazzoni (Free Electrons)

Using Buildroot for a Real Project

PDF

Rajesh Lal (Nokia)

Qt Quick: The Most Advanced UI Technology for Mobile

Talk was canceled

Day 1, 5:15pm

Tim Bird (Sony)

Status of Embedded Linux BoFs

Status-of-Embedded-Linux-2011-10-ELCE.pdf

Lauro Ramos Venancio (Instituto Nokia de Technologia) & Samuel Ortiz (Intel)

The Linux NFC Subsystem

PDF

David Anders (Texas Instruments)

Board BringUp: LCD and Display Interfaces

PDF ODP Resource Page

Day 2, 10:15am

Grant Likely (Secret Lab)

Device Tree Status Report

Laurent Pinchart (Ideas on Board)

Success Story of the Open Source Camera Stack: The Nokia N9 Case

PDF

Avinash Mahadeva & Vishwanth Sripathy (Texas Instruments)

SOC Power Management - Debugging and Optimization Techniques

PDF

Tia Cassett (Qualcomm) & Mike Chalupa (BSquare)

Android Development with the Snapdragon Processor

Day 2, 11:15am

Rafael J. Wysocki (Faculty of Physics, U. Warsaw/SUSE Labs)

Power Management Using PM Domains on SH7372

PDF

Sascha Hauer (Pengutronix e.K.)

A Generic Clock Framework in the Kernel: Why We Need It & Why We Still Don't Have It

PDF

Ruud Derwig (Synopsys)

Android Platform Optimizations

PDF

Tia Cassett & David Brown (Qualcomm)

Kernel Development Using the Dragonboard with the Snapdragon Processor

Day 2, 3:00pm

Inki Dae (Samsung Electronics)

DRM Driver Development For Embedded Systems

PDF

Lorenzo Pieralisi (ARM Ltd.)

Consolidating Linux Power Management on ARM Multiprocessor Systems

PDF

Thomas Petazzoni (Free Electrons)

Using Qt For Non-Graphical Applications

PDF

David Anders (Texas Instruments)

PandaBoard Workshop: Booting the PandaBoard

Day 2, 4:00pm

Marek Szyprowski & Kyungmin Park (Samsung)

ARM DMA-Mapping Framework Redesign and IOMMU Integration

PDF

Keerthy Jagadeesh & Vishwanath Sripathy (Texas Instruments)

Thermal Framework for ARM based SOCs

Marc Titinger (ST Microelectronics)

Efficient JTAG-Based Linux Kernel Debugging

PDF

David Anders (Texas Instruments)

PandaBoard Workshop: PandaBoard Expansion I/O

Day 2, 5:00pm

Tsugikazu Shibata (NEC & Linux Foundation Board Member)

Toward the Long Term Stable Kernel Tree for The Embedded Industry

PDF

Lisko Lappalainen (MontaVista Software)

Secure Virtualization in Automotive

Jeff Osier-Mixon (Intel)

Yocto Project Community BoFs

Luca Coelho (Texas Instruments)

PandaBoard Workshop: WLAN Kernel Hacking with PandaBoard

Day 3, 10:15am

Andrea Gallo (ST-Ericsson)

ARM Linux Kernel Alignment & Benefits for Snowball

PDF

Liam Girdwood & Peter Ujfalusi (Texas Instruments)

Smart Audio: Next-Generation A SoC For Smart Phones

Pawel Moll (ARM Ltd)

Linux on Non-Existing SoCs

Day 3, 11:15am

Koen Kooi (The Angstrom Distribution)

Integrating Systemd: Booting Userspace in Less Than 1 Second

PDF

Sylvain Leroy & Philippe Thierry

Grsecurity in Embedded Linux Used in Android

PDF

MyungJoo Ham (Samsung))

Charger Manager: Aggregating Chargers, Fuel Gauges and Batteries

PDF

Day 3, 2:30pm

Frank Rowand (Sony)

How Linux PREEMPT_RT Works

PDF

Catalin Marinas (ARM Ltd.)

Linux Support For the Large Physical Address Extensions

PDF

Jim Huang (0xlab)

Build Community Android Distribution and Ensure the Quality

PDF

Day 3, 3:45pm

Till Jaeger (JBB Rechtsanwalte)

The Case AVM v. Cybits: The GPL and Embedded Systems

Darren Hart (Intel)

Tuning Linux For Embedded Systems: When Less Is More

PDF

Wolfram Sang (Pengutronix e.K.)

Developer's Diary: It's About Time

PDF

Workshops

Workshops

Presenter(s)	Session Description	Presentation
Chris Simmonds, Freelance Embedded Linux Consultant	Workshop 1: Outside the Box: An Introduction to Embedded Linux and Hardware Interfacing Using the Snowball Board	tar.gz
Karim J. Yaghmour	Workshop 2: Embedded Android Workshop	

Instructions for Presenters

Please create a link in the table for your presentation, copying the style of other links. (You may need to create an account in order to edit the wiki or upload files.)

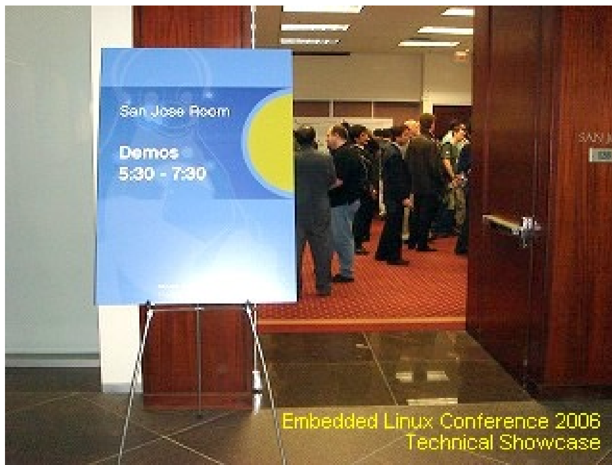
When you have created the link, click on it to upload the file containing your slides.

Categories:

- ELCE
- 2011
- Events
- Presentations

> From: eLinux.org

ELCE 2011 Technical Showcase



Contents

Embedded Linux Conference Europe 2011 Technical Showcase

We are happy to be planning a technical showcase again this year at Embedded Linux Conference Europe. This Wiki page will guide you in giving your demonstration. For conference details, please follow this link to the conference website:

- https://events.linuxfoundation.org/events/embedded-linux-conference-europe



Details

The demo session will be on Thursday, October 27th, between 12:00 and 14:30 in the main event hall foyer. See the conference schedule.

The demo room will be available starting at ***10:00 am*** for showcase demo setup.

Objective

"Seeing is believing!" Unlike in Desktop or Server applications of Linux, in embedded systems Linux technology is usually unseen. Although it is a challenge, we want to expose Linux software and techniques so that other developers who are interested in this technology can learn, and become potential contributors themselves. We hope that through opening up technical details and projects (in the way of Open Source development), these demonstrations will provide insights that are helpful to developers.

Criteria

- No *purely* commercial demonstrations.
- The Embedded Linux Conference Europe should not be used for a pure sales pitch.
- You can show off a product, if there are features or aspects of the product you are displaying that are open source.
- Demonstrations should be of software that is available under GPL or LGPL compatible licenses.
- You will be sharing the hall with sponsors and other paying exhibitors, so your exhibition should be of free and open source technology, tools or techniques.

Coordinator

Tim Bird - Sony Network Entertainment

- e-Mail : tim (dot) bird (a) am (dot) sony (dot) com
- Please replace (a) to "@" and (dot) to "."

What will be prepared by the Linux Foundation

The Linux Foundation will provide the following **FREE OF CHARGE** :

- A Demonstration desk, which is half of a table If you need special arrangements, please contact the coordinator.



- Power supply TBD: EU standard
- Assuming the power is less than 750W. If your demonstration will exceed that, please consult the coordinator.
- A large-format printed poster, based on the file you submit.
- Please note that a dedicated (wired) Internet connection will **NOT** be prepared. However, wireless Internet may be available via the hotel wireless network. We cannot, however, guarantee Internet access.
- Logistics Support
- Details will be given to the demonstrators

Dimension of the desk

- Please refer to the attached photo.

Poster

All demonstrators are requested to prepare a poster to explain the technical outline of the demonstration. You can download the template from this page. CELF will print them out on large size paper. The document should be sent to the coordinator by October 11th.

- ELCE2011_PosterForm.odp
- ELCE2011_PosterForm.ppt

Caution: Do not change the paper size. It is intentionally set to a big size.

Some tips

The demo consists of a poster (printed by LF), with a few sentences talking about a tool, technology or distribution and a table top where you can place anything from:

1. nothing at all, just have a person sitting there to talk to people about what's on the poster, or ...
2. something on a laptop - like a program running or a even a static chart showing some result
3. something running on a board or a device
4. something on an actual CE product

You can also put some flyers there for people to grab, if you'd like.

If you can put charts and graphs on the poster, that's nice but it is NOT required. Neither the poster nor the hardware (if any) needs to be elaborate or require lots of setup. One year I showed up with a camcorder and just talked to people about it's bootup time (which I had worked on).

If you are a presenter, the demo session is a good chance to talk to people one-on-one about your session topic. This gives you more time for interactive Q&A, and a chance to show something hands-on, if that applies to your subject.

How to apply

1. Send an e-mail to the coordinator with the following information:
    i. The contact person, the company name or the project name, the outline of the demonstration.
    ii. The coordinator will acknowledge your application by e-mail.
    iii. Please send all proposals by October 14th, 2011
        i. Apply early, since space is limited!!
2. Prepare the poster and send the data to CELF Office and the demonstration coordinator by **October 19th**. It will be printed out by the conference organizers on big size paper.
    i. Prepare the demonstration equipment
    ii. You can bring the equipment yourself, or ship it.
    iii. If you do not have a consignee which will be convenient for you, send them to the following address: *Address not available yet*

FAQ

- **Q**: Is this opportunity limited to CE Workgroup and/or Linux Foundation members?
- **A**: No. You do not have to be a forum member to have a demo at the conference. However, each demo should be associated with a paid conference attendee.
- **Q**: Is the demonstrator required to make a presentation at the conference?
- **A**: No, this is not required. However, we believe there will be many people who will to be interested in details of the technology, so we hope that you can have a session. Please note that the due date for a session proposal is earlier than the due date for demo sign-up.
- **Q**: Should I assign some dedicated staff to carry on the demonstration?
- **A**: No, this is not necessary. There will be a keynote talk during the demo session slot, which you may want to attend. You may end your demo at the end of the lunch break (at 13:30) or secure your equipment if you wish to attend the keynote. The showcase will officially end at 14:30, at the end

of the break after the keynote.

- **Q**: Is there any fee to demo something in this session?
- **A**: No, it is completely free. However, the session is limited to conference attendees, so please register for the conference if you would not otherwise do so. If you have someone coming who will ONLY run or assist with your demo, and not attend any sessions, then that person may be admitted for free, for just the demo session (and setup and take down). However, each demo should be associated with at least one registered attendee.

Category:

- ELCE

> From: eLinux.org

ELCE Europe 2012 Presentations

Presenters, Demo-ers, Participants: Thanks very much for your participation in Linux Foundation's Embedded Linux Conference Europe 2012.

This page is for collecting the presentations that were made at the conference. During and after the conference we will collect materials from the presenters and place them here. Please watch this page if you are interested in a particular presentation - and if it doesn't show up, please send me and email and we'll try to track it down.

Contents

Videos

Once again, through the diligent work of the Free Electrons team, all videos for ELCE2012 can be found at Videos.

Instructions

Presenters: Please post your technical conference presentations on this page. (See Instructions below the tables)

Table of Presentations

NOTE: If you add a wikilink to your presentation and attempt to upload it via the link, it may fail. If it does, use the Special:Upload page to upload your file.

Keynotes

Keynotes

Presenter(s)	Session Description	Presentation
Mark Shuttleworth, Founder at Canonical	Advancing the User Experience	-
Dave Engberg, CTO Evernote	Why Evernote Runs Their Own Linux Servers Instead of "The Cloud"	-
Brian Stevens, CTO and VP Worldwide Engineering at Red Hat		
Jonathan Corbet, Editor at LWN	The Kernel Report	
Matt Locke, Texas Instruments	Are we headed for a complexity apocalypse for embedded SoCs	PDF
Catarina Mota, Founder at openMaterials	Research into open hardware	
Linux Creator Linus Torvalds and Intel's Open Source Technologist, Dirk Hohndel	Linux: Where are we going?	

Presenters

Presentations

Presenter(s)

Session Description

Presentation

Day 1, 10:10am

Matt Ranostay

Beaglebone: The Perfect Telemetry Platform?

PDF

Jim Huang, 0xlab

Implement Checkpointing for Android

PDF

Wolfram Sang, Pengutronix e.K.

Maintainer's Diary: Devicetree and Its Stumbling Blocks

PDF

Day 1, 11:05am

Matthias Brugger, ISEE 2007 S.L.

A War Story: Porting Android 4.0 to a Custom Board

PDF

Kishon Vijay Abraham

USB Debugging and Profiling Techniques

PDF

Alan Ott, Signal 11 Software

Wireless Networking with IEEE 802.15.4 and 6LoWPAN

PDF

Day 1, 1:20pm

João Paulo Rechi Vita, INdT

Bluetooth Smart devices and Low Energy support on Linux

PDF

Peter Stuge

OpenOCD: Hardware Debugging and More

PDF

Alessandro Rubini

PF_ZIO: Using Network Frames to Convey I/O Data and Meta-Data

PDF

Day 1, 2:15pm

Joo-Young Hwang, Samsung

A New File System Designed for Flash Storage in Mobile

PDF

Alexandre Belloni, Adeneo Embedded

Boot Time Optimizations

PDF

Philipp Zabel, Pengutronix e.K.

Modular Graphics on Embedded ARM

PDF

Day 1, 3:30pm

Karim Yaghmour, Opersys

Inside Android's User Interface

PDF

Samuel Ortiz, Intel

Near Field Communication with Linux

PDF

Arnout Vandecappelle, Essensium/Mind

Upgrading Without Bricking

PDF

Day 1, 6:15pm

Tim Bird, Sony Network Entertainment

BoFs: Developer Tools and Methods: Tips & Tricks

PDF

Nithya Ruff & Ruud Derwig

BoFs: Is HW Availability a Gating Item for Your Software Development

PDF

Elizabeth Flanagan, Intel

BoFs: Yocto Project & OpenEmbedded Community

PDF

Alexandre Belloni

BoFs: The Need For a Fast Bootloader

PDF

Day 2, 10:10am

Sascha Hauer, Pengutronix e.K.

Barebox Bootloader

PDF

Benjamin Zores, Alcatel-Lucent

Dive Into Android Networking: Adding Ethernet Connectivity

PDF

Jiyoun Park, Samsung

Experiences as an OEM with Development of UI Frameworks

PDF

Day 2, 11:05am

Keshava Munegowda, Texas Instruments

FFSB and IOzone: File system Benchmarking Tools, Features and Internals

PDF

Chris Simmonds, 2net Limited

The End of Embedded Linux (As We Know It)

PDF

Steven Rostedt, Red Hat

Understanding PREEMPT_RT (The Real-Time Patch)

ODP

Day 2, 1:20pm

Klaas van Gend, Vector Fabrics

Application Parallelization for Multi-Core Android Devices

PDF

David Anders, Texas Instruments

Board Bringup: You, Me, and I2C

PDF - Resource Page

Rama Pallala, Intel

Linux Power Supply Charging Subsystem

PDF

Day 2, 2:15pm

Agusti Fontquerni, ISEE 2007 S.L.

Embedded Linux RADAR Device

PDF

Matt Porter, Texas Instruments

What's Old Is New: A 6502-based Remote Processor

PDF

Thomas Petazzoni, Free Electrons

Your New ARM SoC Linux Support Check-List

PDF

Day 2, 3:30pm

Tracey M. Erway, Intel & Nithya A. Ruff, Synopsys

Can You Market an Open Source Project?

PDF

Lars Knoll

Qt on Embedded Systems

PDF

Koen Kooi, Circuitco

Supporting 200 Different Expansionboards: The Broken Promise of Devicetree

PDF

Day 2, 4:25pm

Wolfgang Mauerer, Siemens

Àndroit: Real-Time for the Rest of Us

PDF

Anna Dushistova

Eclipse and Embedded Linux Developers: What it Can and Cannot Do For You

PDF

Dave Stewart, Intel

Yocto Project Overview and Update

PDF

Day 3, 10:40am

Vineet Gupta, Synopsys

ARC Linux: From a Tumbling Toddler to a Graduating Teen

PDF

Laurent Pinchart, Ideas on Board

DRM/KMS, FB and V4L2: How to Select a Graphics and Video API

PDF

Frank Rowand, Sony Network Entertainment

Practical Data Visualization

PDF

Day 3, 11:35am

Marcin Juszkiewicz, Linaro

ARM 64-Bit Bootstrapping with OpenEmbedded

PDF

Wim Decroix, TPVision

Practical Experiences With Software Crash Analysis in TV

PDF

Mark Brown, Wolfson Microelectronics

Regmap: The Power of Subsystems and Abstractions

PDF

Day 3, 1:50pm

Wolfgang Mauerer, Siemens

Low-Level Linux Debugging Without Grey Beards

PDF

Hans Verkuil, Cisco Systems

Video4Linux: Current Status and Future Work

PDF

Holger Behrens, Wind River

Yocto Layer for In-Vehicle Infotainment

PDF

Day 3, 2:45pm

Tero Kristo, Texas Instruments

Debugging Embedded Linux (Kernel) Power Management

PDF

Martin Bis, BIS

Real-Time Linux in Industrial Appliances

PDF

Jens Georg, Openismus GmbH

Rygel: Open Source DLNA, ready for Customer Products?

PDF

Day 3, 3:40pm

Yoshitake Kobayashi, Toshiba

Improvement of Scheduling Granularity for Deadline Scheduler

PDF

Tsugikazu Shibata, NEC

LTSI (Long-Term Stable Initiative) Status Update

PDF

Thomas Gleixner, Linutronix

UBI Fastmap

PDF

Instructions for Presenters

Please create a link in the table for your presentation, copying the style of other links. (You may need to create an account in order to edit the wiki or upload files.)

When you have created the link, click on it to upload the file containing your slides.

Categories:

- ELCE
- 2012
- Events
- Presentations

From: eLinux.org

ELC Europe 2007 Presentations

Presentations

Person	Session Description	Presentation
Francois Audeon	Detection & Resolution of Real Time Issues Using TimeDoctor	Detection-of-RT-issues-with-TimeDoctor.pdf video
Tim Bird	BOF: State of Embedded Linux BOF	ELCE-BOF_State_of_Embedded_linux.pdf
Tim Bird and Satoru Ueda	Keynote: CE Linux Forum: the Past, Present and Future	CE_Linux_Forum-Ueda-Bird.pdf
Gustavo Sverzut Barbieri	Fancy and Fast GUIs on Embedded Devices	Fancy_and_Fast_GUIs_on_Embedded_Devices.pdf video
Hugh Blemings	arch/ppc, arch/powerpc and Device Trees - A Walk Through a Port	video
Shane Martin Coughlan	Free Software, Licensing and Business Processes	FSFE-presentation-ELCE-03-11-2007.pdf video
Carsten Emde	Introduction to the Open Source Automation Development Lab (OSADL)	CELF-OSADL.pdf
Jörn Engel	Introduction to LogFS	PDF not available video
Nils Faerber	Overview of LiPS Mission Statement, Architecture and Roadmap	not available
Zhang Feng, Zhao Chunlei, Chen Deyong, Meng Yan	Tutorial: Linux As Handheld Device OS in Lenovo	not available
Harald Fernengel	Qtopia for Developers	not available
Holger Freyther	WebKit on Linux and How It Compares to Other Open Source Engines	ELCE2007_WebKit.pdf video
Thomas Gleixner	Status Overview of Real-Time	video
Thomas Gleixner	Kernel Summit Report	video
Mark Gross	Power Management BoF	not available
Mark Gross	Power Management Quality of Service	not available
Takanari Hayama	Writing DirectFB gfxdriver For Your Embedded System	elce2007_directfb_gfx.pdf video
Marcel Holtmann	Integration of Bluetooth in Embedded Devices	not available
Alexey Korolev	Improving JFFS2 RAM Usage and Performance	Media:ELC-E-JFFS2_RAM_Usage_impr.ppt video
Matthew Locke	OpenEmbedded for Commercial Development	not available
	A Power Management	

Matthew Locke	Architecture For Mobile Devices	mlocke-elce2007-pmarch.pdf
Hiroyuki Machida	Linux for Cell/B.E. and PS3, Related Open Source Projects BoF	20071102-ELCE-Cell-PS3-BoF-E.pdf
Paul Mundt	Asymmetric NUMA: Multiple-Memory Management For The Rest of Us	ELCE2007-Asymmetric_NUMA.pdf
Hadi Nahari	Trusted Secure Isolation For Embedded Linux	ELC-E_SecureIsolation.pdf
Hadi Nahari	Linux Security: Carrier Grade Moving 3G Networks Forward	not available
Sampo Nurmentaus	Creating Cross Platform Multimedia Applications: Case Embedding a Mozilla Based Browser	Sampo-Nurmentaus-Cross-Platform-Linux.pdf
Michael Opdenacker	Linux Tiny - The Diet Must Go On	linux-tiny.pdf
Tsutomu Owa	RT Patch for Celleb - Patch Status and Performance Measurements	ELCE2007_RTpatchForCelleb.pdf
Matt Porter	Methods to Protect Proprietary Components in Device Drivers	Proprietary_Components_In_Device_Drivers.pdf
Manas Saksena	Fedora on ARM Platforms	not available
Gene Sally	How GCC Works, an Embedded Engineers Perspective	not available
Gene Sally	Tutorial: Getting Started with Embedded Linux	not available
Gene Sally	Tutorial: Using RPM as Your Build Environment	not available
Rus Sani	Experiences Using Linux In Carrier Grade Telecom Equipment on Control and Data Plane	Media:srus_elce2007.zip
Frank Scholz	Coherence, an open source DLNA/UPnP framework	CELF-E_2007_Coherence_slides.pdf
Dodji Seketeli	The PokyLinux Distribution: Mobile GNOME at Your Fingertips	poky.pdf
Satoru Ueda and Tim Bird	CE Linux Forum: the Past, Present and Future	CE_Linux_Forum-Ueda-Bird.pdf
Klaas de Waal	Linux in TV, Going From Prototype To Product	LinuxOnTv520.pdf
Wookey	YAFFS	yaffs.pdf video
Wookey and Neil Williams	Tutorial: Embedded Debian Workshop	not available
Vitaly Wool	Parallelizing Linux boot on CE Devices	par.pdf video
Vitaly Wool	Linux Suspend-to-Disk Objectives for Consumer Electronic Devices	std.pdf video
Siarhei Yermalayeu	Linux Clock Management Framework	ELC_2007_Linux_clock_fmw.pdf
Fuxin Zhang	A High Performance Linux-based Home Server Design	not available

Tim Bird	distribution	video

Categories:

- Presentations
- ELCE
- 2007
- Events

> From: eLinux.org

ELC Europe 2008 Presentations

Presentations

- ELCE Videos page.

Sessions

Person	Session Description	Presentation
Mike Anderson	Using a JTAG in Linux Driver Debug	JTAG_In_Linux_Driver_Debug_Anderson.pdf
Mike Anderson	Understanding and Using SMP/Multicore Processors	Understanding_And_Using_SMP_Multicore_Processors_Anderson.pdf
Gustavo Sverzut Barbieri	Rich GUI without pain	Rich_GUI_without_pain.pdf
Tim Bird	Tools and Techniques for Reducing Bootup Time	Tools-and-techniques-for-reducing-bootup-time.pdf Tools-and-techniques-for-reducing-bootup-time.ppt
Vitaly Bordug	Device tree and Embedded Linux	vitb-ELCE-2008-presentation.pdf
Andrew Christian	Handhelds Mojo - Building and Running Ubuntu Distributions on ARM	not available
Shane Coughlan	Strategic Implementation of Free Software in Business	shane-coughlan-presentation-ELCE-07-11-2008.odp
Jake Edge	Avoiding Web Application Flaws In Embedded Devices	elce-2008.pdf elce-2008.odp LWN page
Bas Engel	Digital TV with Linux	Digital_Television_With_Linux.pdf
George France, Brian Avery, & Andrew Christian	Handhelds Mojo: Building and running Ubuntu distributions on ARM	Mojo_CELF_Nov2008.pdf
Klaas van Gend & Ned Miljevic	Building Embedded Userlands	ELCE2008-Building_embedded_user_lands.vGend.Miljevic.pdf
Klaas van Gend	The ELC Europe 2008 end game: Linux Fortunes	Opentux page
Peter Griffin	Porting uClinux to a new architecture	Porting_uClinux_CELF2008_Griffin.pdf
Armijn Hemel	Abusing Universal Plug and Play	elce-presentatie.pdf
Marcel		

Holtmann		
Perry Ismangil & Benny Prijono	PJSIP: Open Source Compact SIP and Media Stack	pjsip_at_Embedded_Linux_Conference_Europe_2008.pdf
Mischa Jonker	Power management on an ARM11 platform	MischaJonker_ARM11_power_management_CELF_ELC_2008.pdf
Denis Oliver Kropp	Open Integration Layer - DirectFB 2.0	not available
Jaya Kumar	Deferred IO and E-Paper Display	E_paper_Displays.pdf
Vasileios Laganakos	Portability and Optimizations of GNU Applications for ARM Embedded Linux	ARM_EmbeddedLinux_Apps_Port.pdf
Philip Lougher	Overview of SquashFS filesystem	squashfs-elce.pdf
Eugeny S. Mints	Taking Linux Power Management to Production Quality	not available
Denis Mishin	A Corner-to-Corner Approach for Cost-Effective Implementation of Consumer Electronics Human Machine Interfaces	not available
Michael Opdenacker	Update on filesystems for flash storage	flash-filesystems.pdf flash-filesystems.odp
Thomas Petazzoni	Choosing embedded graphical libraries	choosing-embedded-graphical-libraries.pdf choosing-embedded-graphical-libraries.odp
Gregers Petersen	Embedded Magic, or How People Suddenly Find Out That They Are Collaborating (Some Thoughts Parsed Through the Brain of an Anthropologist)	not available
Matt Porter	Managing NAND Flash to Optimize Product Longevity	managing_nand_flash_elce.pdf
Bill Roman	Using Appropriate Wear-leveling to Extend Product Lifespan	Datalight_ELC_Presentation_6_Nov_2008.pdf
Frank Rowand	Adventures in real-time performance tuning, part 1	adventures_in_real_time_performance_tuning_part_1-no_hidden.pdf
Frank Rowand	Adventures in real-time performance tuning, part 2	adventures_in_real_time_performance_tuning_part_2-no_hidden.pdf
Frank Scholz	Building bridges - coherence, a DLNA/UPnP framework	not available
David Woodhouse	Community and Embedded Linux	dwmw2-community_and_embedded_linux.pdf

Wookey	Solar hot water geekery: making infinitely versatile home heating controllers with free software and open hardware	hotwaterballoon-CELF2008.pdf src.tgz
Vitaly Wool	Using "Dot Clock" Displays In Embedded Linux Devices	dotclock.pdf dotclock.odp
Vitaly Wool	NAND Chip Driver Optimizaton and Tuning	nand_opt.pdf, nand_opt.odp

Categories:

- Presentations
- ELCE
- 2008
- Events

> From: eLinux.org

ELC Europe 2009 Presentations

Videos of most talks are available at: Free Electrons ELCE 2009

Presentations

Keynotes and Panel

Presenter(s)	Session Description	Presentation
Jon Masters	Porting Linux	Masters-PortingLinux.odp Masters-PortingLinux.pdf
Philippe Gerum	State of Real-Time Linux: Don't Stop Until History Follows	gerum-elce-09.odp gerum-elce-09.pdf

Sessions

Session Presenter(s)	Presentation	File Links
Carmelo Amoroso	LKM Fast Loader Based on ELF Hash Table	C_AMOROSO_Fast_lkm_loader_ELC-E_2009.pdf
Jean-Pierre André, Szabolcs Szakacsits	Unexpected Emergence of Wide Use of NTFS in CE Devices	andre-NTFS3G.pdf Andre-NTFS3G.ppt
Gustavo Sverzut Barbieri	Canola Application and Framework for Rich GUI	GustavoSverzutBarbieri-elce2009-canola2.pdf
Patrick Bellasi	Constrained Power Management	Bellasi-ConstrainedPowerManagement.pdf
Gilad Ben-Yossef	The Good, The Bad and Ugly: On Threads, Processes and Co-Processes	Ben-Yossef-GoodBadUgly.odp Ben-Yossef-GoodBadUgly.pdf
Tim Bird	Analyzing Kernel Function Execution with Ftrace	Bird-Ftrace.pdf Bird-Ftrace.ppt
Vladislav Buzov	Digital TV and Application Store, Solving Security Problems	Buzov-SMACK.pdf
Grégory Clement	How We Got a 3D Application Booting in 5 Seconds Under Linux	Clement-Boot3DApplicationIn5s.pdf
Bas Engel	Accelerating Digital Television Innovating -- Joint SPACE Initiative	2009_ELC_Accelerating_Digital_Television_Innovation.pdf
Florian Fainelli	OpenWrt, as Rapid Embedded Systems Prototyping Framework	fainelli-openwrt-elce2009.pdf
Pierre Ficheux	Using QEMU for Industrial Embedded Applications	pficheux_elce09.pdf
Adriaan de Groot	Software Licensing - A Lot Like Programming	DeGroot-SoftwareLicensing.pdf
Sascha Hauer, Marc Kleine-Budde	U-Boot-v2	Hauer-U_BootV2.pdf
	e2factory - Open Source	

	Embedded Linux Build System	
Cedric Hombourger	Why OpenEmbedded Proved a Good Foundation for MontaVista	Hombourger-Why_oe_good_foundation_for_mv.odp Hombourger-Why_oe_good_foundation_for_mv.pdf
Marcin Juszkiewicz	Hacking with OpenEmbedded	Juszkiewicz-HackingWithOpenEmbedded.pdf
Guennadi Liakhovetski	Embedded Video Capture Under Linux: The Soc Camera Framework	soc-camera.pdf
Bruno Cardoso Lopes	The LLVM MIPS and ARM Back-ends	Lopes-LLVM.pdf
Michael Opdenacker	Update on Boot Time Reduction Techniques	opdenacker-boot-time.pdf opdenacker-boot-time.odp
Samuel Ortiz	Linux Wifi Solutions for Mobile Platforms	Ortiz-elce-2009.pdf
Nicolas Palix, Julia Lawall, Gilles Muller	Coccinelle: A Program Matching and Transformation Tool for Systems Code	Coccinelle_ELC-E_2009.odpCoccinelle_ELC-E_2009.pdf
Pascal Pellet	Linux Embedded Applications in Machine Vision	pascalPellet-e2vELC-E.pdf
Matt Porter	Mythbusters: Android	Mythbusters_Android.pdf
Frank Rowand	A Survey of Linux Measurement and Diagnostic Tools	survey_of_linux_measurement_and_diagnostic_tools.pdf
Alessandro Rubini	Use of the Fast IRQ (FIQ) in ARM-Linux	0910-elce-fiq.pdf
Wolfram Sang	Developer's Diary: The Device Tree	Sang-DiaryDeviceTree.pdf
Stefan Schwarzer	Disko v1.6 – An Application Framework for Embedded Devices	TBD: legal status of 1 slide is unclear
Robert Schwebel	Customizing Embedded Linux Systems with PTXdist	Schwebel-Customizing_with_PTXdist.pdf
Jean-Marc Temmos	Genivi Alliance : An Effort to Build a Linux-based In Vehicle Infotainement Platform	Temmos-GeniviVisteon.pdf
Francesco Virlinzi	A Generic Clock Framework Implementation	ELC_E_2009_Generic_Clock_Framework.pdf
Alex de Vries	Technical Features and Components of Open Source Build Systems	not available
Nina Wilner	Porting Android to Power Architecture	Android_On_Power.pdf
Vitaly Wool	Using Device Trees on ARM Platforms	vwool-device_trees_arm.odp vwool-device_trees_arm.pdf

Birds-Of-A-Feather Sessions

BOF Presenter(s)	Presentation	File Links
Tim Bird	Android	Bird-Android-BOF.pdf Bird-Android-BOF.ppt
Peter Korsgaard, Thomas Petazzoni	Buildroot	buildroot-bof.pdf
Yann E. Morin	Building Our Own Toolchains For Our Embedded Projects: Why, and How To	Morin-Crosstool-NG.pdf Morin-Crosstool-NG.odp
Michael Opdenacker	Small Business	opdenacker-small-business-bof.pdf opdenacker-small-business-bof.odp
Pierre Pronchery	Hackable Devices: The New Possibilities of Open Hardware	not available

Categories:

- 2009
- ELCE
- Events
- Presentations

From: eLinux.org

ELC Europe 2010 Presentations

Presenters, Demo-ers, Participants: Thanks very much for your participation in CELF's Embedded Linux Conference Europe 2010.

This page is for collecting the presentations that were made at the conference. During and after the conference we will collect materials from the presenters and place them here. Please watch this page if you are interested in a particular presentation - and it if doesn't show up, please send me an e-mail and we'll try to track it down.

Contents

Videos

Videos from the conference were recorded by Free Electrons. They are available in full HD and are encoded with the new Open Source and royalty free VP8 audio codec.

Instructions

Presenters: Please post your technical conference presentations on this page. (See Instructions below the tables)

Table of Presentations

Keynotes

Presenter(s)	Session Description	Presentation
Ralf Baechle	Embedded Linux - The State of the Nation	
Ari Rauch	The Dynamic Role of Open Linux Architectures in Today's Mobile Landscape	Elce2010-ari-rauch.pdf

Presentations

Presenter(s)	Session Description	Presentation
Andrew Murray	The Right Approach to Minimal Boot Times	RightApproachMinimalBootTimes.pdf
Andrey Fedotov	Linux Application in Safety-Critical Environment: A Real-Life Example	LinuxInSCEnvironment.pdf
	Eclipse and Embedded Linux	

Anna Dushistova	Developers: What It Can and What It Cannot Do For You	AnnaDushistova.pdf
Armijn Hemel	Introducing the Binary Analysis Tool	Elce2010.odp
	Practical Testing of Open Source Embedded Systems	PDF, Editable (odp)
Arun Raghavan	PulseAudio In The Embedded World	Slides (pdf)
Benjamin Gaignard	Android and GStreamer	Android_and_GStreamer.ppt
Benjamin Zores	State of Multimedia in 2010's Embedded Linux Devices	State_of_Multimedia.pdf
Carmelo Amoroso and Rosario Contarino	Lightweight Prelinker for Kernel Modules	LKM_Preresolver_ELC-E_2010.pdf
David Anders	Board Bringup: Methods and Utilities	PDF ODP
Frank Rowand	Identifying Embedded Real-Time Latency Issues: I-Cache and Locks	Rt_latency_cache_and_locks.pdf
Grant Likely	ARM Flattened Device Tree Status Report	ELCE-2010 ARM Device Tree Status Report.pdf
Gustavo F. Padovan	The Linux Bluetooth Stack	Bluetooth_stack.pdf
Hans Verkuil	Supporting SoC Video Subsystems in Video4linux	SoC_and_V4L2.odp
Harald Welte	Running your own GSM+GPRS network using OpenBSC, OsmoSGSN and OpenGGSN	OpenBSC.pdf
Iago Toral Quiroga	Grilo: Integrating Multimedia Content in Applications	Grilo.pdf
Jake Edge	Understanding Threat Models for Embedded Devices	Edge-ELCE-2010.pdf - Edge-ELCE-2010.odp
Jean-Paul Saman	Porting VLC to TI DaVinci	slides
	A Gentle Introduction to Autotools	slides - demo code
Kevin Hilman	Runtime Power Management	slides
Klaas Van Gend	Deflating the Virtualization Hype in 3 Simple Steps	slides
Koen Kooi	The State of OpenEmbedded and Tooling to Make Life Easier	PDF
Leif Lindholm	Software Considerations When Using High-Performance Memory Systems	Software_implications_memory_systems.pdf
Michael Opdenacker	Flash Filesystem Benchmarks	PDF, ODP
Peter Korsgaard	Do More With Less - On Driver-less Interfacing with Embedded Devices	Do_more_with_less.pdf
Philippe Robin	Facilitating Open Source Development and Collaboration	ELCE-2010-Linaro.pdf
Ray Kinsella	Xen in Embedded Systems	Xen_in_Embedded_Systems.pdf
Robert Schuster (with kind support by Xerxes Ranby for demonstration)	OpenJDK for Embedded Linux Devices	Cross-compiling_OpenJDK.pdf
Robert Schwebel and Sascha Hauer	Barebox: Booting Linux Fast and Fancy	Booting-Linux-Fast-and-Fancy.pdf

Ruud Derwig and/or Mischa Jonker	Portability Is For People Who Cannot Write New Programs - Experience with GNU, LINUX, and other Open Source on ARC Processors	ELC-E_Arc_Linux.pdf
Stefan Kost	Meego Multimedia	MeeGoMultimedia.pdf
Tim Bird	Android System Programming - Tips and Tricks	Android-tips-and-tricks-2010-10.pdf
Wolfram Sang	Developer's Diary: Supporting Maintainers	ELCE10-SupportingMaintainers.pdf
Wookey	YAFFS Updates	yaffsupdate-ELCE-2010.pdf
Yann E. Morin	Crosstool-NG, A Cross-Toolchain Generator	ELC-E 2010 - crosstool-NG.odp
Yoshitake Kobayashi	Linux Kernel Acceleration for Long-term Testing	PDF

Short Sessions

Presenter(s)	**Session Description**	**Presentation**
John Ogness	IPL+UBI: Flexible and Reliable with Linux as the Bootloader	ipl_and_ubi.pdf
Martin Michlmayr	Adapting Debian Installer to NAS and Other Consumer Devices	michlmayr-debian-on-nas.pdf
Ravi Sankar Guntur	A Simple Method to Detect Memory Leaks and Buffer Overruns	SafeMem-ELC-E-2010.pdf
Vitaly Wool	Porting Legacy Code to Linux Userspace Driver Framework	
Will Newton	Exploiting On-chip Memories in Embedded Linux Applications	Will_newton-elc2010-slides.pdf

Birds-of-a-Feather Sessions

Presenter(s)	**Session Description**	**Presentation**
David Anders - Jayabharath Goluguri	OMAP3/4 BoF	OMAP BoF Slides
Frank Scholz	Android and Its Impact On Home Entertainment and Home Automation	
Grant Likely	Small Business Owners BoF	

Tutorial workshops

Presenter(s)	Session Description	Presentation	Broken out presentations
Chris Simmonds	The Embedded Linux Quick Start Guide	linux-quick-start.tar.gz	• 01-linux-quick-start.pdf • 02-linux-quick-start.pdf • 03-linux-quick-start.pdf • linux-quick-start-lab-notes.pdf
Chris Simmonds	What Else Can You Do with Android?	android-inside.tar.gz	• 01-android-inside.pdf • 02-android-inside.pdf • 03-android-inside.pdf • android-inside-lab-notes.pdf

Instructions for Presenters

Please create a link in the table for your presentation, copying the style of other links or as follows:

[[Media:name_of_your_presentation.pdf | name_of_your_presentation.pdf]] Note the supported mime types on the Upload file page. The latter example uses a PDF example, your file type can be different.

(You may need to create an account in order to edit the wiki or upload files.)

When you have created the link, click on it to upload the file containing your slides.

Categories:

- ELCE
- 2010
- Events
- Presentations

> From: eLinux.org

ELC Europe 2013 Presentations

Presenters, Demo-ers, Participants: Thanks very much for your participation in Linux Foundation's Embedded Linux Conference Europe 2013.

This page is for collecting the presentations that were made at the conference. During and after the conference we will collect materials from the presenters and place them here. Please watch this page if you are interested in a particular presentation - and if it doesn't show up, please send me an email and we'll try to track it down.

Contents

Videos

Some of the videos are available at http://www.youtube.com/playlist?list=PLbzoR-pLrL6oxnDyb7IvnNOOBur7z_8tE.

Instructions

Presenters: Please post your technical conference presentations on this page. (See Instructions below the tables)

Table of Presentations

Presentations

Presenter(s)

Session Description

Presentation

Day 1 (24th Oct. 2013)

Chris Simmonds

Keynote: A timeline for embedded Linux

PDF

Michael Christofferson

A Portable Clock Cycle Based Performance Measurement System

PDF

Darren Hart

How Not to Write x86 Platform Drivers

PDF

Dave Stewart

How to Beat Codesweats with the Yocto Project

PDF

Hisao Munakata

LTSI: Long Term Stable Kernel and it's Testing

PDF

Thomas Petazzoni

Device Tree for Dummies

PDF

Russ Dill

Extending the swsusp Hibernation Framework to ARM

PDF

Wolfram Sang

Maintainer's Diary - We Have a Scaling Problem

PDF

Michael Opdenacker

Small Business BOF

PDF

Marek Vasut

Using i.MX28/i.MX233 without Freescale tools

PDF

Scott Wood

TPL:SPL Loading SPL (and, SPL as just another U-Boot config)

PDF

Stefano Babic

Falcon Boot: current status and enhancements

PDF

Mark Rutland

Devicetree: The Disaster So Far

PDF

Mischa Jonker

Fighting latency: How to optimize your system using perf

PDF

Paul Eggleton

Keeping It Green: Integrated QA with the Yocto Project

PDF

Simon Glass

Verified Boot on Chrome OS and How to do it yourself

PDF

Lukasz Majewski

Device Firmware Upgrade (DFU) - present situation and future development

PDF

Gregory Clement

Common Clock Framework how to use it

PDF

Will Deacon

From Weak to Weedy: Effective Use of Memory Barriers in the ARM Linux Kernel

PDF

Jonathan Austin

Linux From Sensors to Servers - When is Linux not Linux?

PDF

Patrick Titiano

Use-Case Power Management Optimization: Identifying & Tracking Key Power Indicators

PDF

Jagan Teki

U-Boot Verified RSA Boot Flow on ARM (With Demo Run)

PDF

Simon Glass

Driver Model, Kconfig and little Patman

PDF

Tom Hacohen

EFL - A UI Toolkit Designed for the Embedded World

PDF

Marta Rybczynska

Going Linux on Massive Multicore

PDF

Matt Ranostay

Sigrok: Using Logic to Debug Logic

PDF

Tim Bird

Status of Embedded Linux (October 2013)

PDF

Karim Yaghmour

Android Platform Debugging and Development

PDF

Behan Webster

LLVMLinux: The Linux Kernel with Dragon Wings

PDF

Jessica Zhang

Modernize Embedded Linux Software Development Tool to Achieve Development Anywhere

PDF

Day 2 (25th Oct. 2013)

Morten Rasmussen

BOF Power Aware Scheduling

PDF

Alison Chaiken

Best Practices for Long Term Support and Security of the Device-Tree (DT)

PDF

Martin Bis

Secure Embedded Linux Product - A Success Story

PDF

Jeff Osier-Mixon

BOF Yocto Project & OpenEmbedded

PDF

Nicolas Launet

Linux Kernel Debug and Profiling Tools

PDF

Arnout Vanadacappelle

Android on Non-Mobile Embedded Systems

PDF

Christian Babeux

Bridging the Gap Between Hardware and Software Tracing

PDF

Holger Dengler

Linux Secured Integrity - Protention Against Remote Attacks

PDF

Peter Korsgaard

Buildroot: What is new

PDF

Mehmet Faith Karagoz

Node.JS Appliances on Embedded Linux Devices

PDF

Guillene Ribiere

Optimize DMA Configuration in Encryption Use Case

PDF

Karim Yaghmour

Running Code in the Android Stack

PDF

Yoshitake Kobayashi

An Essential Relationship between Real-time and Resource Partitioning

PDF

Pantelis Antoniou

Board File to Device Tree Migration: A War Story

PDF

Lucas Stach

Next-Generation DMABUF : How To Efficiently Play Back Video on Embedded Systems

PDF

Frank Rowand

Using and Understanding the Real-Time Cyclictest Benchmark

PDF

Pawell Moll

BoF Hardware Trace in the Kernel

PDF

Koen Kooi

Are Embedded Build Systems Still Needed?

PDF

Attila Kinali

Debugging Electronics for the Software Engineer

PDF

Mark Hatle

SPDX and the Yocto Project

PDF

Sascha Hauer

Barebox and Bootloader Specification

PDF

John Hawley

Building Interesting and Complex Robotics Using x86 Computers

PDF

Scott Garman

Building Robots That Can See

PDF

Instructions for Presenters

Please create a link in the table for your presentation, copying the style of other links. (You may need to create an account in order to edit the wiki or upload files.)

When you have created the link, click on it to upload the file containing your slides.

Categories:

- ELCE
- 2013
- Events
- Presentations

> From: eLinux.org

ELC Europe 2014 Presentations

Presenters, Demo-ers, Participants: Thanks very much for your participation in Linux Foundation's Embedded Linux Conference Europe 2014.

This page is for collecting the presentations that were made at the conference. During and after the conference we will collect materials from the presenters and place them here. Please watch this page if you are interested in a particular presentation - and if it doesn't show up, please send me an email and we'll try to track it down.

Table of Presentations

NOTE: If you add a wikilink to your presentation and attempt to upload it via the link, it may fail. If it does, use the Special:Upload page to upload your file.

Sessions

Presentations

Session Description

Presenter(s)

Presentation

Day 1, 11:15am

The Orc Quest for Better Embedded Multimedia Performance Adding MIPS support to liborc

Guillaume Emont, Igalia

PDF

Enhancing Real-Time Capabilities with the PRU

Ron Birkett, Texas Instruments

PDF

Performance Analysis Using the Perf Suite

Mans Rullgard

The MIPS Creator CI20 Developer Board: Firing Up the Community & Melting Servers

Ian Oliver, Imagination Technologies

Is SSH Really Secure?

Peter Tornberg, Fox Technologies

How to Collaborate on Linux Kernel Development

Mauro Carvalho Chehab, Samsung

Day 1, 12:15am

12 Lessons Learnt in Boot Time Reduction

Andrew Murray, Embedded Bits Limited

PDF

Case Study: Building a High Quality Video Pipeline Using GStreamer & V4Linux on an i.MX6

Sean Hudson, Mentor Graphics

PDF

Overcoming Obstacles to Contributing to Linux

Tim Bird, Sony Mobile

PDF

What Can Possibly Go Wrong?

Konrad Zapalowicz, Cybercom Poland

Day 1, 2:30pm

Bluetooth Low Energy and Internet of Things

Marcel Holtmann, Intel

Open Source Medical Accessories

Philip Verbist, XsOnline Rocky De Wiest, Belgian Defence

Transactional Device Tree & Overlays: Making Reconfigurable Hardware Work

Pantelis Antoniou, NVIDIA

PDF

Coming Soon, an Open Source Project Near You - the Linaro LNG Open Data Plane Initiative

Michael Christofferson, Enea

Day 1, 3:30pm

A Double-Agent Developer: ARM vs x86

David Anders, CircuitCo

PDF

Choosing your System C Library

Khem Raj, Comcast

High Performance NFV Infrastructure (NFVI): DPDK Host Applications with Neutron/OpenStack and VNF Acceleration

Vincent Jardin, 6WIND

Tutorial: Setting up ktest.pl - Embedded Edition

Steve Rostedt, Red Hat

PDF

Day 1, 4:30pm

Pairing WebKit and Wayland for Linux-Based Embedded Web Content Presentation Systems

Žan Doberšek, Igalia

PDF

Redundant booting with U-Boot

Thomas Rini, Texas Instruments

PDF

Software Defined Storage: Changing the Rules for Storage Architects

Ric Wheeler, Red Hat

Day 1, 5:30pm

BoFs: Coreboot

Moderated By Ronald G. Minnich, Google

BoFs: What are the Technical Issues on Linux For IoT

Shinsuke Kato, Panasonic Bryant Eastham, Panasonic

BoFs: Yocto Project / OpenEmbedded

Tracey Erway, Intel

BoFs: TPM Subsystem

Peter Huewe

BoFs: First Failure Data Capture for Linux

Michael Holzheu

Michael Müller, IBM

Day 2, 9:00am

Embedded Android Workshop

Karim Yaghmour, Opersys

PDF

Day 2, 11:15am

Building Tools From The Outside In: Bringing User-Centered Design to Embedded Linux

Belen Barros Pena, Intel

PDF

Introduction to Skia: A Modern 2D Graphics Library

Eduardo Lima, Igalia

PDF

The DRM/KMS Subsystem From a Newbie's Point of View

Boris Brezillon, Free Electrons

PDF

Using Embedded Linux for Infrastructure Systems

Yoshitake Kobayashi

Use "strace" to Understand Linux

Harald König, Bosch-Sensortec GmbH

Day 2, 12:15am

Chromium OS Audio System

Dylan Reid, Google

PDF

prpl Foundation / OpenWrt Panel

Kathy Giori, Qualcomm Atheros Felix Fietkau, OpenWrt Mathieu Olivari, Qualcomm Atheros Luka Perkov, OpenWrt

Tuning Android for low RAM

Chris Simmonds

u-root: A Go-Based binutils Providing Scripting Convenience and Compiled-Program Performance

Ronald G. Minnich, Google

PDF

Kernel Hacking for Hobbyists - An Outsider's Perspective

Manuel Schölling

Linux Performance Tools

Brendan Gregg, Netflix

Testing Video4Linux Applications and Drivers

Hans Verkuil, Cisco

Day 2, 2:30pm

Compressing Strings of the Kernel

Wolfram Sang

Demystifying Android's Security Underpinings

Karim Yaghmour, Opersys

PDF

Embedded GPUs: A Case For Open Source Drivers

Lucas Stach, Pengutronix

Generic PHY Framework: An Overview

Kishon Vijay Abraham, Texas Instruments

PDF

Square Pegs in Round holes, or System Level Performance Data and perf

Pawel Moll, ARM

x86 Instruction Encoding and Nasty Hacks We Do in the Linux Kernel

Borislav Petkov, SUSE

Day 2, 3:30am

Supporting a New ARM Platform: The Allwinner Example

Maxime Ripard, Free Electrons

PDF

Systemd for Embedded Linux - Challenges and Opportunities

Michael Olbrich, Pengutronix

PDF

Two years of ARM SoC Support Mainlining: Lessons Learned

Thomas Petazzoni, Free Electrons

PDF

USB and the Real World

Alan Ott

PDF

UserModeLinux Status Report

Richard Weinberger, Sigma Star GmbH

Where is My Crystal Ball?

Daniel Lezcano, Linaro

Day 2, 4:30am

Cycle Accurate Profiling With Perf

Pawel Moll, ARM

PDF

EFL - A UI Toolkit Designed for the Embedded World

Cedric Bail, Samsung

PDF

LTSI: Status and Plans For Long-Term Stable Kernel

Tsugikazu Shibata, NEC

Hisao Munakata, Renesas

PDF

Real Safe Times in the Jailhouse Hypervisor

Jan Kiszka, Siemens

PDF

Automated Linux Kernel Crash Infrastructure - Eye In the Digital Sky

Igor Ljubuncic, Intel

Day 3, 11:15am

Fast Boot: Profiling and Analysis Methods and Tools

Christopher Hallinan, Mentor Embedded

Introduction to prpl Foundation

Art Swift, prpl Foundation

Leveraging Open-Source Power Measurement Standard Solution

Patrick Titiano

PDF

Using Linux Throughout the Complete UAV Stack

Koen Kooi, Linaro

PDF

Ftrace Kernel Hooks: More Than Just Tracing

Steven Rostedt, Red Hat

Using Persistent Memory Effectively

Matthew Wilcox, Intel

Day 3, 12:15am

High-Speed Data Acquisition With the Linux I/O Framework

Lars-Peter Clausen, Analog Devices

PDF

Porting Linux to a New Architecture

Marta Rybczynska, Kalray

PDF

ACPI And Device Trees - Friends Or Foes?

Rafael J. Wysocki, Intel OTC

Automatic NUMA Balancing

Rik van Riel, Red Hat

Stateless Systems, Factory Reset, Golden Master Systems and systemd

Lennart Poettering, Red Hat

Day 3, 2:30pm

devicetree: Kernel Internals and Practical Troubleshooting

Frank Rowand, Sony Mobile

PDF

Mastering the DMA and IOMMU APIs

Laurent Pinchart, Renesas

PDF

Software Update in Embedded Systems

Stefano Babic, DENX

PDF

How to Design a Linux Kernel API

Michael Kerrisk, man7.org

Scaling Userspace @ Facebook

Ben Maurer, Facebook

Day 3, 3:30pm

Buildroot: A Deep Dive Into The Core

Thomas Petazzoni, Free Electrons

PDF

rtmux: A Thin Multiplexer To Provide Hard Realtime Applications For Linux

Jim Huang, ITRI

PDF

Secure and flexible boot with U-Boot Bootloader

Marek Vašu, DENX

PDF

First Glimpse at Shingled Drives

Hannes Reinecke, SUSE Labs

Day 3, 4:30pm

BoFs: Device Tree Next Steps

Grant Likely, Linaro

Tame the USB Gadgets Talkative Beast

Krzysztof Opasiak, Samsung

PDF

Tizen-Meta as Security and Connectivity Layers For Yocto Project

Dominig ar Foll, Intel

PDF

Linux Kernel Tinification

Josh Triplett, Intel

Systematic Testing of Fault Handling Code in Linux Kernel

Alexey Khoroshilov, Russian Academy of Sciences

Day 3, 5:30pm

Closing Game

Tim Bird, Sony Mobile

Instructions for Presenters

Please create a link in the table for your presentation, copying the style of other links. (You may need to create an account in order to edit the wiki or upload files.)

When you have created the link, click on it to upload the file containing your slides.

Categories:

- ELCE

- 2014
- Events
- Presentations

> From: eLinux.org

ELC Presentations

North America

- ELC_2006_Presentations
- ELC_2007_Presentations
- ELC_2008_Presentations
- ELC_2009_Presentations
- ELC_2010_Presentations
- ELC_2011_Presentations
- ELC_2012_Presentations
- ELC_2013_Presentations
- ELC_2014_Presentations
- ELC_2015_Presentations

Europe

- ELC Europe 2007 Presentations
- ELC Europe 2008 Presentations
- ELC Europe 2009 Presentations
- ELC Europe 2010 Presentations
- ELCE Europe 2011 Presentations
- ELCE Europe 2012 Presentations
- ELC Europe 2013 Presentations
- ELC Europe 2014 Presentations

Categories:

- ELC
- ELCE

> From: eLinux.org

Embedded Developer BoF 2010



This is an event for embedded Linux enthusiasts together with the community developers beside an international Linux event, LinuxCon Japan.

Contents

September 28th (Tue) at Roppongi Hills

- From 5:30pm
  - 40th Floor (Roppongi Academy Hills) directory
- Everyone who is interested in embedded Linux will be welcomed
  - No admission fee required
- Light meal will be served

Co-located with LinuxCon 2010

- about LinuxCon 2010

Last Year



- Thomas Glixner made briefing of the latest situation of real-time preemption of Linux.

Talks

- 5:30
  - Welcome talk and introduction of CE Linux Forum
- 5:45
  - Summary of CE Linux Forum technical activities
- 6:15
  - Linux and OSS technical talks
    - "Android on Ubuntu for developer" (Tetsuyuki Kobayashi / KMC)
      - Slides
      - Please check previous Japan Technical Jamboree
  - (If you wish to talk something, contact Satoru.Ueda(a)jp_dot_sony_dot_com)
- 6:50
  - CE Linux Forum, the story from the origin to the future
    - S. Ueda (CELF Marketing Group Chair / Sony)
- 7:30
  - Let me introduce my talk in LinuxCon Japan!
    - lightening Session
- 8:00
  - **DRAWING!**
    - **You have chance to win some fancy goods!** (Sponsored by Sony Corporation)

Categories:

- 2010
- Events

> From: eLinux.org

Embedded Linux Conference 2009

The was CE Linux Forum's 5th annual embedded Linux conference was held in San Francisco, California, April 6-8, 2009.

See http://www.embeddedlinuxconference.com/elc_2009/index.html

Presentations from the conference are posted at: http://tree.celinuxforum.org/CelfPubWiki/ELC2009Presentations

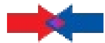
Video from the conference can be found here courtesy of Free Electrons: Video Presentations Free Electrons has been kind enough to provide some of these presentations in High Definition, making it possible to more easily read slides directly from the video.

Categories:

- Events
- 2009
- ELC

> From: eLinux.org

Embedded linux events

	This page is may need to be merged with other page(s) including Events. Please help to merge the articles, or discuss the issue on the talk page.

Contents

Links to conferences web sites

Here is information about Embedded Linux-related events (conferences and symposiums)

The main conferences each year are:

- Ottawa Linux Symposium (OSL)
  - See http://www.linuxsymposium.org/2008/schedule.php
- LinuxConf Australia (LCA)
  - See http://www.linux.org.au/LCA
- Linux Kongress
  - See http://www.linux-kongress.org/2008/
- FOSDEM
  - See http://www.fosdem.org/2008/
- Embedded Linux Conference (ELC)
  - See http://www.embeddedlinuxconference.com/
- Real-time Linux Workshop (RTLWS)
  - See http://www.realtimelinuxfoundation.org/

Miscellaneous other conferences:

- LinuxWorld - embedded track?
  - See http://www.linuxworld.com/events/
- LinuxConf Europe??
- SCALE
  - See http://socallinuxexpo.org/
- Japan Technical Jamboree
  - See http://tree.celinuxforum.org/CelfPubWiki/JapanTechnicalJamboree
- Korean Technical Jamboree
  - See http://tree.celinuxforum.org/CelfPubWiki/KoreaTechJamboree3
- Plumber's conference
  - See http://linuxplumbersconf.org/
- Vendor conferences - are presentations available??
  - MontaVista Vision
    - See http://www.mvista.com/vision/

- ARMdev
- other semiconductor conferences??

Embedded Systems Conference India 2010

- ESC India 2010

Links to conference proceedings

Here are some links to various conference proceedings:

PRESENTATIONS

- All Topics

OLS

- All the OLS papers from 2002-2007 - http://kernel.org/doc/ols

FOSDEM (embedded track)

- Fosdem 2005 embedded kernel papers
- Fosdem 2006 embedded kernel papers

Other Linux events (which might have content relevant for embedded

Linux Foundation Collaboration Summit 2008

Videos of the Linux Foundation Collaboration Summit plenary sessions are now available for viewing on our website, as are interviews with some of our attendees.

Link: https://www.linux-foundation.org/events/video/gallery --- Several of the workgroups have also posted slides and/or summaries from the summit as well.

Link: https://www.linux-foundation.org/events/node/53

Categories:

- NeedsMerge
- Events

> From: eLinux.org

Embedded Linux Summit 2010

Here is information about the Embedded Linux Summit, 2010.

This page has some information about the location, date, time, attendees, sponsors and other plans for this event.

See Embedded Linux Mini Post Summit 2010 for information about the follow up meeting at ELC Europe.

Contents

Attendees

Here is a table of confirmed (or semi-confirmed) attendees for the summit:

Name	Company	Notes
Tim Bird	Sony	Lead organizer of the summit
Bdale Garbee	HP	
Matt Mackall	Selenic	Embedded Maintainer
Greg Ungerer	Snapgear	uClinux Maintainer
Paul Mundt	Renesas	SH Maintainer
Hisao Munakata	Renesas	
Ralf Baechle		MIPS Maintainer
David Woodhouse	Intel	Embedded Maintainer
Grant Likely	Secret Lab	Device tree maintainer
I.P. Park	Samsung	*(tentative)*
Doewan Kim	LGE	
Yoshitake Kobayashi	Toshiba	
Brian Swetland	Google	*(tentative)*
Magnus Damm	Renesas	
Bryan Huntsman	Qualcomm Innovation Center	MSM maintainer
Dirk Hohndel	Intel	Chief Linux and OpenSource Technologist *agreed to come, but is now on sabbatical, and I can't reconfirm*
Arjan van de Ven	Intel	Kernel developer *(tentative)*
Shinsuke Kato	Panasonic	

Attendee Guidelines

Meeting Details

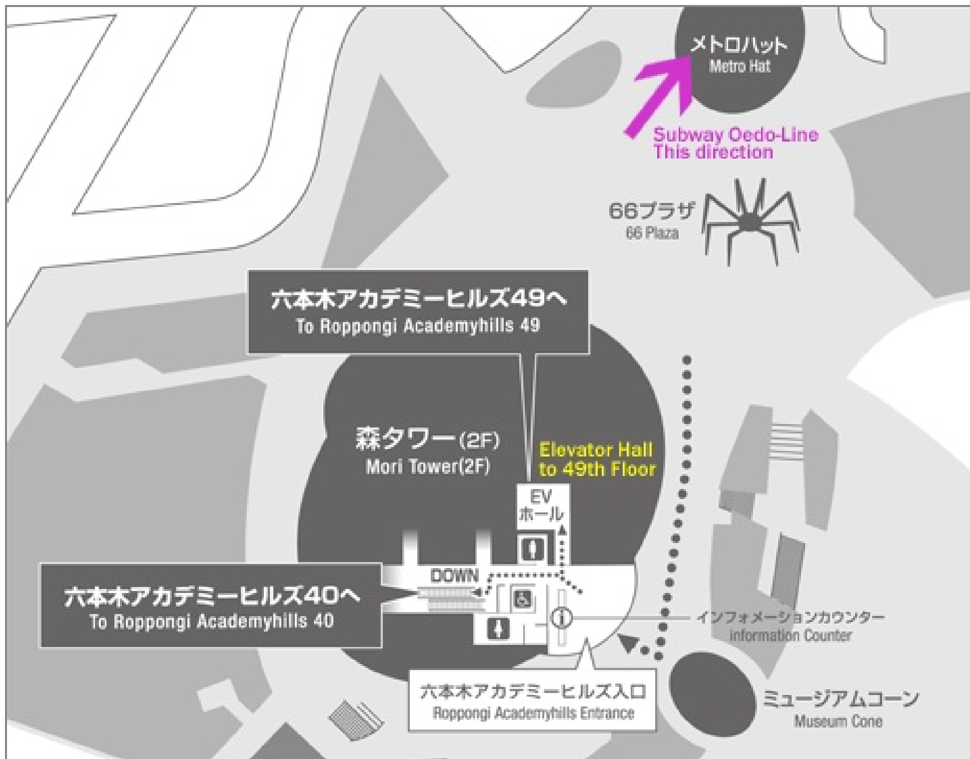
- Date: September 30, 2010
- Time: 9:00 am to 5:00 pm
- Location: Tokyo, Japan
- Venue: Roppongi Academy Hills
  - 49th Floor of Roppongi Hills Building
  - Conference Room 5
  - Map: http://www.academyhills.com/aboutus/map.html

Lunch will be provided.

http://www.academyhills.com/english/aboutus/mission_and_vision/index.html

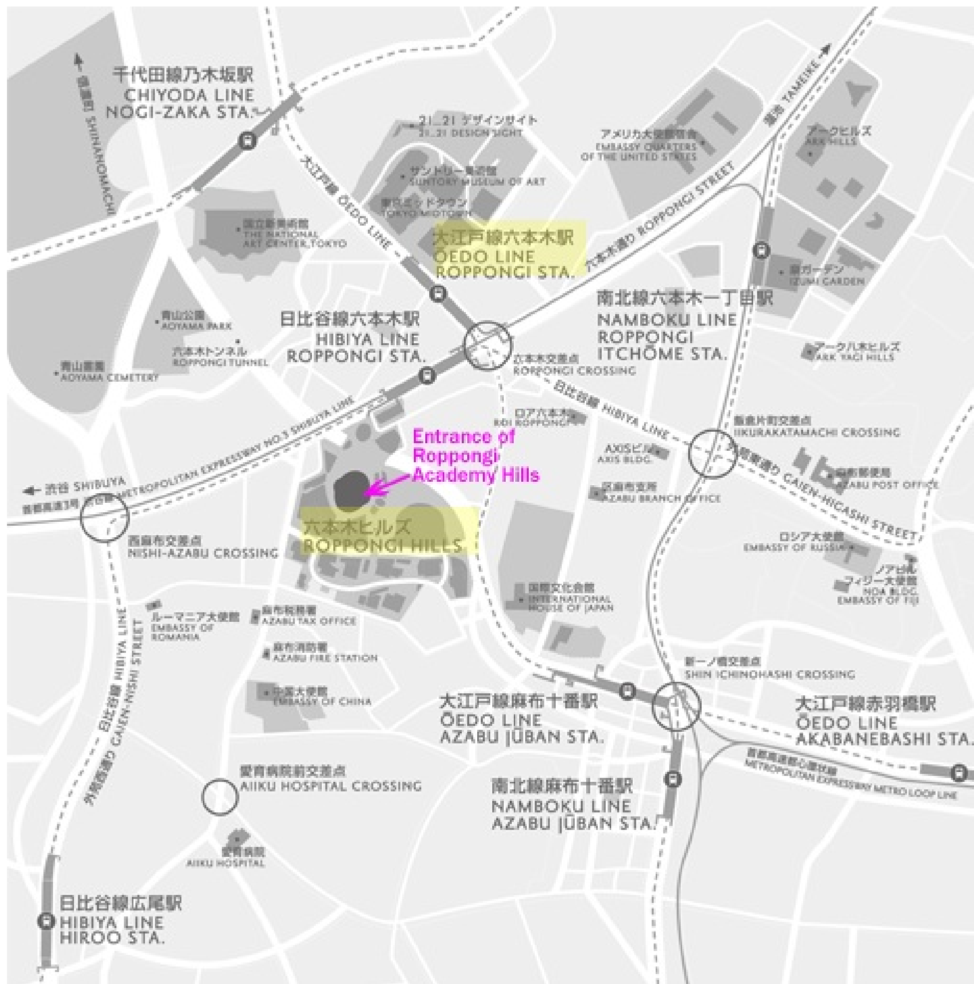
Directions

-
    - Directions: From Shiodome station (next to Park Hotel)
        - Take the Toei Oedo line to Roppongi Hills



-
    -
        - Please be noted that the direction guide from the subway station is assuming you

use the "Tokyo Metro, Roppongi Station" not the "Toei (Municipal) Oedo Line, Roppongi Station". The attached wide area map indicates the location of Oedo Line Roppongi Station, which is about 300m (980ft) from the Tokyo Metro Roppongi station.



Hotel Information

We are recommending the same hotel for the Embedded Linux Summit as for LinuxCon Japan. This is the Park Hotel Tokyo which is located in southeast Tokyo, in the Shimbashi district of the city.

If you are attending LinuxCon, you can use the Linux Foundation Room rate, of 15,000 yen (about $175) per night. See instructions on the LinuxCon Japan Hotel page for making a reservation.

- Hotel web site: http://www.parkhoteltokyo.com/index.html
- Directions to the hotel: http://www.parkhoteltokyo.com/directions/index.html

Agenda

The agenda is being discussed on the summit mailing list.

Here is the draft that Tim proposed for different topics for the meeting:

```
1. mainline gap
  1.1 version gap is most prevalent form
  1.2 how to reduce it?
    1.2.1 can stable release maintenance help with this?
    1.2.2 patch mining - pros and cons
    1.2.3 proxy contributors - pros and cons
    1.2.4 development methods: git vs. git+quilt vs. big honking vendor tarballs!
2. architecture-specific issues (ARM, MIPS, x86)
  2.1 SOC vendor tree isolation
  2.2 defconfigs
  2.3 recent industry initiatives (e.g. Linaro)
3. SMP in embedded
4. Android current status
  4.1 kernel requirements
  4.2 mainline gap
    4.2.1 how it's managed
    4.2.2 plan for closing gap
5. Meego current status (same sub-points as above) (Dirk or Arjan?)
6. WebOS current status (same sub-points as above) (Bdale?)
7. The state of flash file systems
8. embedded power management
  8.1 suspend blockers?
  8.2 runtime PM
9. industry contribution rate - how to increase it
  9.1 impedances to industry participation in open source
    9.1.1 systematically identifying impedances
10. industry and community pain points
  10.1 industry pain points - what do developers spend their time doing?
  10.2 community pain points - what to developers spend their time doing?
  10.3 need a survey or are attendees representative?
11. CELF projects - (Tim - 30 minutes)
  11.1 status of current work
  11.2 ideas for future work
12. Automated testing
  12.1 is it of any use?
  12.2 what kinds are needed
  12.3 should CELF or LF fund something? (long painful history here...)
```

Additions:

```
13. Unified bootloader architecture
14. Device tree status (Grant Likely - ?? minutes)
```

Sponsor

- CE Linux Forum

Mini summit followup meeting in the UK

Russell King won't be coming to the meeting in Tokyo. Since a number of us will be in England at ELC Europe in late October, we've decided to have a brief meeting there as well, to include Russell and continue discussing some of the ideas raised in Tokyo.

See Embedded Linux Mini Post Summit 2010

Categories:

- Events
- 2010

> From: eLinux.org

Event Planning Pages

This page has links to pages used to plan events and summits.

The main use for this pages, after the event has completed, is for templates for future event planning.

Here are the pages:

- Technical Conference 2006 Planning
- Linux PM Summit 2006 Planning
- Small Business Conference Reason obsolete, 2006 is long gone

Category:

- ToDelete

> From: eLinux.org

Events/Kernel Summit 2011 ARM Subarch Maintainership Workshop

\< Events

We're currently planning an ARM Subarchitecture Maintainership Workshop for the first day of Kernel Summit in Prague, October 23, 2011. Watch this page for details, or contact Grant Likely, Nicolas Pitre, or Arnd Bergmann.

There is limited space available for this meeting. Seats were provided on a first-come, first-served basis and we have now surpassed our self-imposed limit of 24 people. However, if you still would like to attend, or would like to nominate someone who you think is an important participant, then email Grant Likely \grant.likely@secretlab.ca\. We may be able to open up additional seats as needed.

We're also extending an open invite to all kernel summit invitees. If you've been invited to kernel summit and you'd like to attend the ARM workshop, then email Grant so that we know you are coming. Otherwise we'll make you sit on the floor.

Contents

Proposed agenda topics

Here is the current list of proposed discussion topics. Feel free to add items to this list.

- arm-soc tree: 6 month retrospective
- DT bindings for GPIO and pin mux
- the pin mux subsystem from linusw (especially if it is still RFC by then)
- progress with the single zImage work
- presentation/status of the DMA and memory management work wrt CMA (some SOC specific hacks should go away once this is available)
- DT porting progress
- boot architecture status
- Board-specific code maintainer (proposed by Igor Grinberg)
- Common method for registering errata (CPU, cache controller etc.) (Proposed by Catalin Marinas)
- CPU Topology DT binding (Proposed by Catalin Marinas)
- Fine grained DMA Coherency (Proposed by Catalin Marinas)
- Handling off-the-tree dependencies of devices (Proposed by Rafael J. Wysocki)
- Per-device PM QoS - user space interaction (Proposed by Rafael J. Wysocki)
- Universal platform drivers (for ACPI, PNP, etc.) (Proposed by Rafael J. Wysocki)
- Suspend vs wakeup events (Proposed by Rafael J. Wysocki)

Minutes

Etherpad notes [1]

Photo



Schedule

The ARM maintainership workshop runs from October 23-25, 2001 in Prague. Sunday October 23rd is dedicated to discussion topics of interest to the ARM subarchitecture maintainers, and October 24 and 25 are organized as a hacking sprint. The workshop is co-located with Kernel Summit and there will be some joint events. You can also refer to the Kernel Summit Schedule. Workshop attendees are welcome to attend any of the Kernel Summit presentations on Tuesday, October 25.

DRAFT SCHEDULE - SUBJECT TO CHANGE

Sunday, October 23

Monday, October 24

Tuesday, October 25

9:00am

Introductions and announcements (Grant Likely)

Hacking Sprint

(ksummit) Report from Monday's session

9:30am

Common Infrastructure Reports

- Pinmux subsystem (Stephen Warren)
- Common Clock (Sascha Hauer)
- CMA (Marek Szyprowski)

(ksummit) Minisummit/workshop reports

10:00am

10:30am

Break

Break

11:00am

Common Infrastructure Reports

- Complex Driver Model Dependencies
- ARM Cortex-A15 Architecture Issues (Catalin Marinas)

Hacking Sprint & Ksummit presentations

11:30am

12:00pm

Lunch

12:30pm

1:00pm

Single zImage (Nicolas Pitre, Deepak Saxena)

Hacking Sprint

Hacking Sprint & Ksummit presentations

1:30pm

2:00pm

Device Tree Migration progess (Grant Likely)

2:30pm

Break

3:00pm

Break

Hacking Spring & Ksummit presentations

3:30pm

ARM SoC Tree; 6 month review (Arnd Bergmann, Nicolas Pitre)

4:00pm

4:30pm

TBD

(ksummit) GPG identity verification

5:00pm

Summary, Action Item review and Hacking Sprint planning

Break

5:30pm

Break

Break

6:00pm

6:30pm

Evening Reception at Hotel

Dinner

Joint Kernel Summit, ELC-E, LinuxCon Europe reception

Hacking Sprint

- *list target goals here*

Delegates

Registration for the ARM Workshop is now closed. Here is the list of registered participants. If you are registered for this event, feel free to fill in your arrival information below if you want to organize ride sharing from the airport.

Confirmed

Name

Roles

Arrival information

Departure

Chris Ball

MMC maintainer and OLPC developer

Ohad Ben-Cohen

OMAP engineer

Arnd Bergmann

arm-soc tree maintainer

DB-Bus-1061 OCT 22 16:18

David Brown

MSM maintainer

BA856 OCT 22 6:50pm

BA855 Oct 29 12:15pm

Mark Brown

ASoC maintainer

LH1402 OCT 22 23:15

LH1393 OCT 29 09:50

Lennert Buijtenhek

Marvell SoC maintainer

KL3125 OCT 22 16:05

KL1358 OCT 28 19:20

Benoît Cousson

OMAP engineer

AF2482 OCT 22 7:50pm (sharing taxi w/Kevin Hilman)

AF1083 OCT 26 7:25

Magnus Damm

shmobile maintainer

Will Deacon

ARM engineer

Nicolas Ferre

Atmel maintainer

AF4314 OCT 22 8:10pm

Thomas Gleixner

random Linux dude

Igor Grinberg

OMAP engineer

OK287 OCT 22 9:15am

OK286 OCT 29 11:55pm

Sascha Hauer

imx maintainer

Kevin Hilman

OMAP power management maintainer

AF2482 OCT 22 7:50pm (sharing taxi w/Benoit Cousson)

AF1083 OCT 29 7:25am

Olof Johansson

Tegra maintainer

BA856 OCT 22 6:50pm

BA855 Oct 29 12:15pm

Kukjin Kim

Samsung SoC maintainer

KE935 OCT 22 17:50

KE936 OCT 29 19:25

Russell King

ARM architecture maintainer

BA856 OCT 22 18:50

BA855 OCT 26 12:15pm (probably share taxi with swarren)

Grant Likely

Device Tree, GPIO and SPI maintainer

AC9294 OCT 22 11:45am

Tony Lindgren

OMAP maintainer

Pavel Machek

Freezer/hibernation maintainer

Catalin Marinas

ARM engineer

Kyungmin Park

Samsung mobile boards & drivers maintainer

Nicolas Pitre

arm-soc tree maintainer

Jean-Christophe Plagniol-Villard

Atmel maintainer

Wolfram Sang

Deepak Saxena

Linaro Kernel WG manager

Marek Vasut

Various pxa boards maintainer

Paul Walmsley

OMAP clock framework maintainer

Stephen Warren

Tegra maintainer

BA856 OCT 22 6:50pm (1 open taxi space)

BA855 OCT 26 12:15pm (probably share taxi with RMK)

Marc Zyngier

ARM engineer

Sitting in from Kernel Summit

Name

Roles

Mike Frysinger

Ben Herrenschmidt

Greg Kroah-Hartman

Nominated, but have not yet responded.

Name

Roles

Barry Song

CSR maintainer

Declined - will not attend

Name

Roles

Eric Miao

pxa/mmp maintainer

Marek Szyprowski

attending V4L2 workshop at same time

Paul Mundt

shmobile maintainer

Linus Walleij

ST-Ericsson SoC maintainer

Haojian Zhuang

pxa/mmp maintainer

Volunteers

- Planning Committee: Arnd Bergmann, Grant Likely, and Nicolas Pitre
- AV:
- Logistics:
- Ride sharing:

Categories:

- Events
- 2011
- KS

> From: eLinux.org

Ftrace Function Graph ARM

This page holds information from Tim Bird's talk at Linux Symposium Montreal (July, 2009), and at ELC Europe and the Japan Linux Symposium (October, 2009).

In Canada, Tim talked about patches against the 2.6.31-rc1 kernel tree for adding function graph tracing to the ARM architecture, for the Ftrace system.

In France and Japan, Tim talked about patches against 2.6.32-rc5.

Contents

Presentation

Here is the presentation from the Japan Linux Symposium (also given at ELC Europe, in Grenoble France)

- Measuring Function Duration with Ftrace (october 2009 update) (PDF)

Here is the presentation from the Canada Linux Symposium session:

- Measuring Function Duration with Ftrace (first version) (PDF)

Paper

Here is Tim's paper for the Canada Linux Symposium for this work:

- Measuring Function Duration with Ftrace Paper (PDF)

Documentation

See the file Documentation/trace/func-duration.txt, after applying the patches.

(Or, if you applied the 2.6.31 patches, see the file Documentation/trace/func-graph.txt However, this version is deprecated.)

Patches

Below are patches for the function graph support and function duration tracer, for multiple kernel versions.

2.6.33 patches

Here are some patches to add support for the function_graph tracer to ARM, and to add support for 'tracing_thresh' to the function_graph tracer.

- Media:Function-graph-ARM-w-thresh-2.6.33-mar-2010.tgz

To apply, extract the patches and the series file. If you have 'quilt', extract the file directly into the top level of the kernel source tree (the top directory is called 'patches'). Then do "quilt push -a".

If you don't have quilt, apply the patches manually, in the order specified in the 'series' file, with a command like: "patch -p1 \<patches/foo.patch"

2.6.32-rc5 patches

- Media:Ftrace-patches-oct-2009.tgz

To apply, extract the patches and the series file. If you have 'quilt', extract the file directly into the top level of the kernel source tree (the top directory is called 'patches'). Then do "quilt push -a".

If you don't have quilt, apply the patches manually, in the order specified in the 'series' file, with a command like: "patch -p1 \<patches/foo.patch"

2.6.31-rc1 patches

The following patches were submitted to the kernel mailing list in early July 2009. There are patches against kernel version 2.6.31-rc1.

- Media:Arm-sched_clock-notrace.patch
- Media:Add-function-graph-tracer-support-for-ARM.patch
- Media:Func-graph-duration-filter.patch
- Media:Optimize-duration-filter-discard.patch

2.6.29 patches

The following patches were developed internally at Sony, but never submitted to mainline. They are posted here in the hopes that they will be useful to someone.

- Media:Add-ARM-function-duration-tracer-2.6.29.tgz

This tar archive contains a patches directory, with a series file and several patch files. To use, extract at the source of your 2.6.29 kernel tree, and apply using quilt with 'quilt push -a'.

Alternatively, apply the patches individually using 'patch -p1 \<patches/p1.patch', for each file listed in patches/series (in the order specified in the file).

This patch set was tested against a stock (mainline, from kernel.org) 2.6.29 kernel, on an OMAP OSK development board (ARM-based, TI OMAP chip).

patches for gcc 4.4.0

Note recent ARM toolchains instrument the code with calls to '__gnu_mcount_nc' instead of 'mcount'. If you get a compiler warning about missing the symbol '__gnu_mcount_nc', you should apply the patch mentioned in this e-mail also: (http://marc.info/?l=linux-arm-kernel&m=124946219616111&w=2)

```
[Thanks to Jean Pihet of MontaVista for pointing this out]
```

Here is the above patch, and a patch which adds support for __gnu_mcount_nc to the function_graph tracer.

- Media:Support-ftrace-with-newer-compilers.patch
  - this is the patch mentioned above
- Media:Gnu_mcount_nc-func_graph.patch

Tools

FDD

The 'fdd' tool is now incorporated into the patch set, and is located in the kernel *scripts* directory after applying the patches. See the documentation in the kernel tree for instructions for use.

FTD

- Media:Ftd.txt - Function Trace Dump - post-trace analysis tool

To install:

```
* Download
* rename to 'ftd': mv Ftd.txt ftd
* make it executable: chmod a+x
* Put it on your path somewhere: sudo mv ftd /usr/local/bin
```

History

Based on feeback from ftrace developers on the kernel mailing list, and from other developers at the tracing mini-summit in Montreal, I started working on an updated duration tracer, using a different filtering approach. This work was completed, and resulted in a new trace, the "function_duration" tracer, being completed in September of 2009.

This work was described at ELC Europe and the Japan Linux Symposium, in October, 2009. I was hoping to mainline the patches before the events, but ran out of time due to bugs on my main testing platform.

Further Work

I need to try to mainline this feature. I'd like to get it into Linux-tip or Linux-next sometime before the 2.6.33 merge window.

If you don't see it in mainline, just send me an e-mail asking about the status.

Category:

- Development Tools

> From: eLinux.org

Geek Cruises

Perl Whirl / Linux Lunacy / MySQL Swell 2004

- http://www.geekcruises.com/top/ll4_top.htm
    - (old) info on the cruise
- http://www.geekcruises.com/ - **Geek Cruises**

Slides

- http://dave.org.uk/perlwhirl/ - Dave Cross
- http://www.ubiqx.org/presentations/
    - Christopher R. Hertel
- http://x1.develooper.com/~robert/pw2004_talks/
    - Robert Spier
- http://www.deepspace6.net/sections/docs.html
    - Mauro Tortonesi

Pictures

- http://flickr.com/photos/davorg/
    - Dave Cross
- http://www.metaphoric-reality.com/galleryng/
    - Grant
- http://www.stonehenge.com/merlyn/Pictures/Trips/2004/04-10-Italy-PerlWhirl4/
    - Randal L. Schwartz
- http://rikers.org/gallery/200410italy
    - Tim Riker
- http://zeus.cira.ca/thumbs - Ervin Ruci
- http://www.ruberg.no/gallery/2004geekcruise
    - Bjørn Ruberg
- http://www.ubiqx.org/~crh/Photos/GeekCruise04/
    - Christopher R. Hertel
- http://x1.develooper.com/~robert/italy/ http://xrl.us/dqtx - Robert Spier

Category:

- Categories

> From: eLinux.org

GStreamer 2010 Presentations

Presenters and Participants: Thanks very much for your participation in GStreamer Conference 2010.

This page is for collecting the presentations that were made at the conference. During and after the conference we will collect materials from the presenters and place them here. Please watch this page if you are interested in a particular presentation - and it if doesn't show up, please send me an e-mail and we'll try to track it down.

Contents

Videos

Videos from the conference were recorded and will be made available as soon. An announcement will be made, and a link placed here, when these are ready.

Instructions

Presenters: Please post your technical conference presentations on this page. (See Instructions below the tables)

Table of Presentations

Presentations

Session Description	Presenter(s)	Presentation
Keynote - GStreamer - Current and future development	Wim Taymans, Collabora Multimedia	
Webkit, HTML5 and GStreamer	Philippe Normand, Igalia	
Cross platform development with GStreamer	Michael Smith, Songbird	
Challenges of video editing in your pocket	Edward Hervey, Collabora Multimedia	
A GStreamer based framework for adaptive streaming applications	Emanuele Quacchio, ST Microelectronics	
GStreamer and OMAP4	Rob Clark, Texas Instruments	
Implementing DLNA using GStreamer	Zeeshan Ali, Nokia	
Integrating VideoConferencing into Everyday Applications	Olivier Crete, Collabora	
Optimizing multimedia with Orc	David Schleef, Entropy Wave.	
Case study - Tandberg and GStreamer	Håvard Graff, Tandberg	
Landell - live streaming for the masses	Luciana Fujii, Holoscopio	
Case study - Flumotion and GStreamer	Zaheer Merali	
Case study - GStreamer on Axis devices	Jonas Holmberg, Axis	
Using ICE middleware with GStreamer to implement real-time QoS-aware video streaming for remotely controlled vehicle.	Andrey Nechypurenko and Maksym Parkachov	
3D Stereoscopic and GStreamer	Martin Bisson	
Case Study - Intel SMD elements in GStreamer	Josep Torra, Fluendo	
WebM and GStreamer	Sebastian Dröge, Collabora Multimedia	
Case study - Using gstreamer for building automated webcasting systems	Florent Thiery, Ubicast.eu	
Interactivity in GStreamer pipelines	Jan Schmidt, Oracle Corporation	

Instructions for Presenters

Please create a link in the table for your presentation, copying the style of other links or as follows:

[[Media:name_of_your_presentation.pdf | name_of_your_presentation.pdf]] Note the supported mime types on the Upload file page. The latter example uses a PDF example, your file type can be different.

(You may need to create an account in order to edit the wiki or upload files.)

When you have created the link, click on it to upload the file containing your slides.

Categories:

- 2010
- Events
- Presentations

> From: eLinux.org

International Technical Jamboree

Contents

International Technical Jamboree

CELF is hosting an International Technical Jamboree on 14th and 15th June in Yokohama Japan. This is a technical conference to share knowledge and experience related to Linux and its use in consumer electronics devices. This is an open conference; all information is shared and any code presented must be available under an Open Source Initiative accepted license (preferably GPL or LGPL).

NOTE: No previous registration required. No admittance fee required

Call for Sessions

- To propose a session topic, please send e-mail to the Jamboree coordinator.
- 9 Slots (80 minutes each) for sessions are planned.
- All sessions should be conducted in English.
- Anyone can propose to present a session.
  - Non-CELF members are eligible to host sessions.

Venue

- At Yokohama Grand Intercontinental Hotel.
- Hotel Homepage in English
- Hotel reservation form : Media:Hotel_Reservation_Form_Y.pdf
- **Please note that the special room price for attendees of this event is available for bookings received before 25th May.**
- Transportation Guide To Intercontinental Yokohama

Time Slots

Day (1) 13th June

Time

Item

Detail/Presentation availability

09:00-10:20

Technical overview of Qt (our C++ GUI/app. framework) and its porting to embedded Linux- by John Ryland (Trolltech)

(not available)

Coffe Break

10:30-11:50

Making Mobile Phone with CE Linux- by Masashige Mizuyama(Panasonic Mobile Communications)

ITJ2005Detail 1-2 presentation available

12:30-13:20

Writing custom gfx_driver for DirectFB- by Katsuya Matsubara(Renesas/iGel)

ITJ2005Detail 1-3 Presentation available

13:30-15:00

Technical Discussion: "Experiences with Open Embedded and the OMAP5912 Starter Kit" - by Steve Johnson(Matsushita/Panasonic)

ITJ2005Detail 1-4 presentation available

Coffe Break

15:10-15:50

Technical Discussion: "Learning the Kernel, and Finding Performance Problems, with KFI"- by Tim Bird (Sony)

ITJ2005Detail 1-5 - presentation available

16:00-17:20

Technical Discussion: "devicelinux.org, a brand new open source effort aiming at creating an easy-to-deploy Linux platform for embedded devices"- by Markku Ursin

ITJ2005Detail 1-6 presentation available

- All sessions on day 1 are open to members and non-members alike.

Day (2) 14th June

Time

Item

Detail/presentation availability

09:00-10:20

Technical Discussion 1: XvFAT File System Development'*Detail: Xv Fat Discussion :)*Presentation Available'***Technical Discussion 2: Fast Boot/Shutdown with Power Management Function**

(not available) **Presentation Available**- by Hiroyuki Machida (Sony)

Coffe Break

10:30-11:50

Technical Discussion: "UHAPI 4Linux, an Open Source implementation of the Universal Home API"- by John Vugts (Philips) and Arjen Klomp (Logica CMG)

ITJ2005Detail 2-2 presentation available

13:00-14:20

Technical Discussion: "Embedded Linux Packaging and Development Toolkit for Consumer Electronics"- by Heung Nam Kim (ETRI)

ITJ2005Detail 2-3 presentation available

Coffe Break

14:30-15:50

Technical Discussion: "Linux Kernel Tracer"- by Toru Nojiri (Hitachi)

ITJ2005Detail 2-4 presentation available

16:00-16:30

Special Session: "Ottawa Linux Symposium and CE Linux Forum"- by Tim Bird (Architecture Group Chair)

ITJ2005Detail 2-5 presentation available

16:30-17:00

CE Linux Forum Plenary Meeting (Forum Members limited)

17:00-19:00

CE Linux Forum, 2nd Anniversary Party (**Non forum members are also welcomed.** Free of charge)

- All sessions on day 2 are open to members and non-members, **except** the session from 16:30 (4:30pm) to 17:00.
- Sorry for my mistake in indicating the dates. The dates are 13th (Mon) and 14th (Tue) which are corrected on 17th May.

Category:

- Events

From: eLinux.org

Japan ESEC 2006



Contents

For the visitors of CELF Booth in the ESEC 2006

- このページはESEC 2006のCELFブースにお立ち寄りいただいた方のために用意致しました。
  - This page is prepared for the visitors of ESEC 2006, one of major wxhibitoin about the embedded system.
  - This page is described in Japanese.
- CE Linux Forumのブースにお立ち寄りいただきまして有り難うございました。このフォーラムは組込み機器に使う観点からLinuxの発展を願い、日夜開発を進めるエンジニアに依るコミュニティーです。今回のESECでは手狭なブースでは有りましたが最近の技術検討の成果をそれぞれの第一人者による解説を試みました。座席の用意も無く心苦しい限りですが、これをご縁にフォーラムにぜひご参加ください。フォーラムの活動の多くは会員でなくとも参加可能です。勿論会員になっていただく事を心から歓迎いたします。

主なフォーラムのイベント

Worldwide Embedded Linux Conference

- 毎年４月にシリコンバレーに世界中の組込み関係リナックス開発者が集まり技術ディスカッションを行います。
  - http://www.celinux.org/elc2006/index.html

Japan Technical Jamboree

- 日本ではほぼ２ヶ月おきに技術カンファレンスを行っています。最先端の開発者も多数参加する中、Linuxを始めとするオープンソースで提供される組込みシステム構築に役立つソフトウエア技術のディスカッション。質疑応答が30分以上続くことも珍しくなく密度の濃い一日です。詳しくは Japan Technical Jamboree Guidance をお読みください。

ESEC2006 ブースプレゼンテーション資料

- こちらの内容に関しては（フォーラム一般紹介も含めて）ご質問はテクニカルジャンボリーコーディネータまでお寄せください。コーディネータの連絡先は下記のWikiページに有ります。「今回キーパーソンがESECに来場できなかったので・・・」「改めてCELFの活動内容の詳細を聞きたいのだが・・・」等も遠慮なくお問い合わせください。歓迎いたします。
    - Japan Technical Jamboree 9

CELFの紹介

- Media:CELFIntro_prnt.pdf

技術解説資料 / 6月28日

ハイバネーション技術（ソニー・神長さん）

- Media:esec2006-snapshot-boot.pdf

Kernel Tracing技術（日立・野尻さん）

セキュアなOS構築（IBM・宗藤さん）

技術解説資料 / 6月29日

システムサイズ計測ツール（NEC・池田さん）

- Media:size_tool_esec_2006-06-29_jp.pdf

Direct FBの改良（ルネサス・宗像さん）

- Media:directfb_improve.pdf

技術解説資料 / 6月30日

CELF検討技術の適用事例(WILLCOMコアモジュール フォーラム・近藤さん)

- Media:ESEC2006-case_study-SGWP.pdf

CABI（CPU Resource Management）（早稲田大学・菅谷さん）

- Media:ESEC_cabi_and_priority_boost_2006_0630.pdf

携帯電話にLinuxを組込む（パナソニックモバイル・水山さん）

Categories:

- Events

- 2006

From: eLinux.org

Japan Jamboree To WELC 2006

Contents

Invitation to Worldwide Embedded Linux Conference

- This is a page to motivate Japanese embedded system developers to attend the World Embedded Linux Conference 2006. Please understand this page is mainly in Japanese.

World Embedded Linux Conferenceへの誘い

- CE Linux Forumは毎年「国際版ジャンボリー」を開催しています。この場には、組込み系の世界中の開発者のみならずメインライン系の開発者なども参加して活発な技術ディスカッションが繰り広げられます。日本からも多数参加して頂けるよう願いつつ、このページを作りました。

Road to Silicon Vallay

アメリカに初めて行く人に

日本からの便

- アメリカ西海岸にはほぼすべての便が日本を午後に出発して同じ日の午前中早めの時間に目的地の空港に着きます。帰りは昼前後の出発で翌日の夕方に到着する便が殆どです。
  - サンノゼ空港が一番便利です。サンノゼ国際空港には成田から直行便が一日一往復あります。アメリカン航空の便で日本航空がコードシェアしています。成田は第一ターミナル発ですので間違わないように。
  - サンフランシスコ空港には成田、中部、関西の各空港から多数便が有ります。サンフランシスコ国際空港からサンノゼ方面は、レンタカーで１時間弱。そのほかバス、またはカルトレインでも行けます。但しカルトレインは土曜、日曜は極端に便数が減ります（一時間に１本程度）。
    - Caltrainのサイト
    - カルトレインの駅はMillbrae Stationが最寄りです。BART（Bay Area Rapid Transit: 地下鉄）で一駅です。サンフランシスコ方面に行く列車はMillbraeには行きません。ご注意を。
    - サンフランシスコ国際空港のサイト

現地の様子

- 治安：　はじめてアメリカに行かれる方は治安を心配されるかと思います。一般にシリコンバレー地区は極めて治安が良い場所です。もちろん保証するわけでは有りませんが、日本とさほど変わらないという印象を持たれている方が多いです。
- クレジットカード：　アメリカではクレジットカードは身分証明書のようなもの。必ず国際カードを用意して行く事

をお勧めします。レンタカーを借りる場合、ホテルのチェックイン等利用シーンは数知れず。またスーパーマーケットなどでもたった数ドルの買い物でもクレジットカードを使う人が多い。

- 交通手段：　レンタカーを借りるのをお勧めします。もちろんアメリカは右側通行。いくつか日本とは異なる交通ルールが有ります。シリコンバレー地域の方は安全運転を心がけている方が多い印象を持たれている方が多いです。もともと車社会のアメリカは非常に高齢の方などもハンドルを握っています。そのような方も居る前提でみなさん運転をされているようです。
  - アメリカと日本の違い
    - 指定が無ければ赤信号でも右折は可です。もちろん安全を確認した上で。
    - 踏切では一時停止しません。
    - 信号の無い交差点ですべての道が「一時停止」となっている場所がかなりあります。その場合は最初に交差点に来て一時停止した車が優先して通過します。
    - 通勤時間に二人以上で乗車していないと使えないレーンがところどころにあります。"Commuter Lane"とか呼ばれるもので例えば101号線のサンノゼ周辺などにあります。パトロールカーが見張っていることが多く、日本からの旅行者が知らずに捕まるケースがあるようです。
  - 国際免許証で運転できますが、必ず日本国内の免許証も同時に携帯する必要があります。
  - ガソリンスタンドはセルフサービスが目立ちます。操作は書いてある通りにすれば良い。但しクレジットカードをポンプ（給油機）に挿入するとZIPコードの入力を求められる事が有ります。日本で作ったカードには勿論ZIPコードは登録されていません。その場合はガソリンスタンドの従業員の指示に従ってください。International Card でZIPがregisterしていない、と、伝えれば、給油中クレジットカードを預かるケースやとにかく給油してしまい後でカウンターに行くケース等が有るようです。
- 公共交通手段：　シリコンバレーでは路面電車網が整備されてきました。LRT (Light Rail Train)と呼ばれる路面電車でサンノゼ空港からかなり広範に移動できます。今回の会場もLRTの停留所がそばに有ります。運営はVTAという公社。但し日本の市電のような頻度での運転では無く、15分おき程度。学生とか年齢の関係、運転免許が無い等でレンタカーが借りられない場合は活用してください。サンノゼ空港からLRTの駅までは無料シャトルバスが走っています。LRTの切符は駅に自動販売機で購入します。Mountain View Station等でカルトレインと乗り換えが出来ます。
  - VTAのホームページ
  - VTA全路線図
    - Santa Clara Convention Center はLight Rail / Mountain View - Winchester線「Great America Station」が最寄り駅。
- 食事・水：　サンノゼにはダウンタウンにジャパンタウンも有り、日本食レストランも多い。また、中華、韓国、ベトナム料理など多彩です。概してどれも分量が多めな傾向が有ります。もちろんステーキハウスとかも有ります。魚介も比較的新鮮です。水道水は飲用可能ですがミネラルウオーターを買って飲む人が多いようです。酒類を購入するときに小柄な日本人は成人である事を確認される事が有ります。そのような場合は顔写真付きで、誕生日の西暦での記載がある、例えば国際免許証などを見せましょう。
- シリコンバレーについて詳しく触れている旅行ガイドブックはあまりありません。「地球の歩き方」等に有ったかと思います。

ミーティングにて

- まさか居ないとは思いますが、ネクタイとかは全く不要。カジュアルな服装で全く問題有りません。ジーンズ、Tシャツでもお構いなし！
  - ちなみにフォーマルウエアが必要なレセプションとかはフォーラム主催のものは有りません。
  - 但しアメリカのコンベンションセンター、映画館等、エアコン（冷房）が猛烈に効いている事が有ります。今回も４月ですから既にそのような季節になっているかも知れません。ウインドブレーカーのようなものを持っていると良いでしょう。ちなみにアメリカの航空会社の機内も大分気温は低めに設定してあるようです。
- 質問等は積極的に。これはアメリカのマナーとも言えるでしょう。セッションの最中に手を挙げる勇気がなかったらセッション終了後にプレゼンターを捕まえましょう。
- 会話中に相手の言っている事が判らなくなったら「もう一度言ってくれ」とお願いするのは（何度でも）全く失礼にはなりません。むしろ判ったのか判らないのかとらえどころのないスマイル（ジャパニーズスマイルと陰口を言われています）は不気味に感じられるようです。
  - 相手の言っていることを会話で理解する手として、自分なりの表現で言い直してみるなどは有効です。
  - 今回会場に居るのは敵対する相手では無いでしょう。外交官では無いのですから、英語表現に過敏になる必要は

有りません。

- ただ単に質問するだけでは無く、「私はこんなコードを開発しているのですが（コードを見せる）」といった展開の方がより一層歓迎されます。どうしてもコミュニケーションにハンディキャップがある場合はあらかじめ自己紹介カードのようなものを用意しておくと良いかも知れません。「ネタ振りカード」の要領で。
  - World Embedded Linux Conference用「ネタ振り」カードのフォームを用意してみました。例えば何枚かプリントアウトして会場で初めてあった方に手渡したりしてはいかがでしょうか。
    - Media:selfIntroduction.ppt

Categories:

- Events
- 2006

> From: eLinux.org

Japan Linux Symposium 2009 for Embedded System Developers

This page is for embedded system developers who are interested in participating Japan Linus Symposium 2009. CE Linux Forum is sponsoring this event as well as the Kernel Summit 2009.



Contents

Call For Presentations

Proposal due date

- **May 1st, 2009**
  - Will be extended to **May 15th**.

Encouragement to submit session proposal

```
As you may know, the due date of the Japan Linux Symposium
(JLS) presentation proposal is May 1st.  Though it is not CELF
event, it must be important one because it is held in East
Asia (Tokyo) where so many embedded system developers are
living.  I wish you to be encouraged to propose your
topics!

* Especially for embedded system developers:
==========================================
The Japan Linux Symposium is held just after the Kernel Summit
which means we can expect the most of core Linux developers
will also remain to attend.  It is definitely a great opportunity
to talk embedded technology stuffs together.  Please try to
take the chance!

You are requested to post:

  - The session title
  - Abstract (150 words)
  - Biography (150 words)

only.

# Don't worry.  You are not required to prepare Japanese text.
# All above should be in English.   ;)

--- Japanese ---

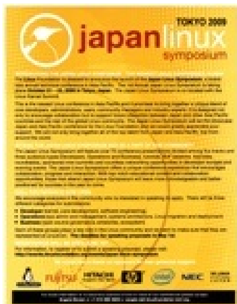

既にご存知の通り、きたる10月21日から東京・秋葉原にてThe Japan
Linux Symposiumが開催されます。こちらはCE Linux Forum主催の
イベントではありませんが、是非皆様の積極的なご参加をお願い
申し上げます

このイベントはKernel Summitに引き続き開催されます。Kernel
Summitには全世界から80名にも及ぶ最先端リナックス開発者が
招待されます。このような方々と直接交流する場としても、JLSは
特筆されます。特に組込み技術はコミュニティ一からも注目を
集めています。

只今プレゼンテーションの募集受付中です。締め切り日は
5月1日です。皆様の積極的な提案を期待しています。
ご自身では些細な事と思われている事が実は大きな意義を
持っている、そのような事もこれまで多数見受けられます。

提案は、150ワードの内容紹介、150ワードの自己紹介とタイトル
だけです。また、このシンポジウムでは論文を提出する必要は
有りません。セッションスタイルも講演型、BoF型、パネル型
など自由に構成できます。
```

Flyer

	• A flyer to promote the session proposal is prepared. ◦ File:JLS2009flier 1.pdf

Categories:

- 2009

- Events

> From: eLinux.org

Japan Technical Jamboree

Table Of Contents

Contents

Introduction

The Japan Technical Jamboree is a forum-wide technical meeting of the CE Linux Forum. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.

このサイトは原則として日本語のみです。 ***THIS PAGE was ;-) JAPANESE LANGUAGE ONLY***

10月**18**日時点で**66**名様のご登録を頂いております。会場定員の都合で申し込みを打ち切らせていただく可能性も有りますので、ご参加予定の方は早めに 登録をお願いいたします。登録ページにアクセス出来ない場合はコーディネータにメールまたは電話でご連絡いただきますようお願いいたします。(66 attendees have registered as of October 18. Entry might be closed due to the capacity of the meeting room. Register. Hurry up If you can not access the registration page, then contact the coordinators by mail or phone.)

第１回　CELF テクノジャンボリー・東京 / The First CELF Techno-Jamboree Tokyo

- コーディネーター：
  - Coordinators
  - 宗像尚郎 / Hisao Munakata
    - munakata.hisao@renesas.com
    - TEL: 03-3861-2642

  - 上田　理 / Satoru Ueda
    - satoru.ueda@jp.sony.com
    - TEL: 03-6409-5011
- 2004年9月29日にMembersメーリングリストに送りました第一回開催のご案内レターはこちらでダウンロードできます。(You can download the First Jamboree announcement message sent on September 29, 2004.)

Date and Location

- **2004**年**10**月**29**日（金）、午前**10**時から午後**4**時まで。(October 29, 2004, Friday, 10:00 through 16:00)
- 場所**(location)**：
  - 東京簡易保険会館（ゆうぽうと・五反田）(Tokyo Easy Insurance Hall "Yuu-port Gotanda")
    - 山手線・五反田駅から徒歩**5**分(5 min walk from Gotanda station, Yamanote-line)
    - 山手線、りんかい線・大崎駅から徒歩**7**分(7 min walk from Ohsaki station, Yamanote-line/Rinkai-line)
    - 東急池上線、大崎広小路駅に隣接（荒天の場合以外は、五反田駅ないし大崎駅から徒歩をお勧めいたします。五反田駅から大崎広小路駅までの距離は僅か300mです）(Adjacent to Ohsaki-Hirokohji station, Tokyu-Ikegami line)
  - ホームページ(Home Page)：http://www.u-port.fukushi.kampo.japanpost.jp
  - 地図(Map)：http://www.u-port.fukushi.kampo.japanpost.jp/accs/accs.html
  - 10/5に定員約100名の部屋を予約しました。(We reserved a room of 100 person capacity on October 5).
- お願い(Please)
  - こちらの会場では商用電源のコンセントが充分に確保できません。恐れ入りますがテーブルタップをお持ちの方はご持参いただければ助かります。(The room does not have enough power outlets. It would be grateful if you attend with your extension cable.)
  - インターネット接続の設備はございません。ロビーではPHSの電波を拾うことが出来ます。(No facility for internet connection. You can catch PHS wave in the lobby.)
- 触れるまでもなく、服装はカジュアルでどうぞ。(Of course, dress code is "casual".)

Registration

- こちらのページで参加登録願います。この登録ページはCELFメンバー登録されていない方はアクセス出来ません。もしCELFメンバー登録をされていない方が登録ページから参加登録される場合は、(Register for the Jamboree at this page. Only registered members of CELF can access this registration page. Otherwise:)
  - どなたかメンバー登録されている方に登録していただく(Ask somebody who is already registerd as CELF member.)
  - または、ご自身でメンバー登録していただく(Or, register yourself as CELF member here.)
  - または、コーディネーターにご連絡願います。(Or, contact the coodinators.)

ショートプレゼンテーション / Short Presentation

- 当日、皆様からショートプレゼンテーションをいただければ幸いです。各社**5**〜**10**分程度を目安。もし各社の中で異なる技術開発項目を持たれている複数のグループが参加される場合は、それぞれグループ毎にプレゼンテーションしていただいても結構です。下記のいずれかの内容でお願いいたします。
  - We appreciate you make a short presentation of 5-10 minutes for each company. If more than one groups with different technical, development issues in one company, each group may make a presentation. Choose one from below:
  - どのような技術分野に関してどのようなパッチをご提供いただけるか
    - what patches you can submit in what technical field?
  - どのような技術分野のいかなる事で開発に難儀しているか
    - what issue you are struggling with?
  - どのような事に関してお互いに情報交換したいか
    - On what topics you would like to exchange information each other?
- ご承知の通り、CELFはGPL/LGPLないしはそれらに互換性があるライセンスによって自由に交換できるソフトウエアをベースに技術ディスカッションを進める場です。上記に関しましてもその条件を前提としてお話しいただければと

思います。
  - As you know, CELF is a place for technical discussion based on software freely exchangeable with GPL/LGPL or compatible licenses. Please talk with that assumption.
- このプレゼンテーションは必須ではありません。またプレゼンテーションの内容に関して一切のお約束等は無くて結構です。No commitment, No obligationで結構です。
  - This presentation is not mandatory. And you do not have to commit anything on the content of your presentation.
- プロジェクターは用意いたします。恐れ入りますがプレゼンテーション用のPCは各位でご準備ください。
  - We prepare a projector. Please come with your own PC for presentation.

Agenda（仮）(tentative)

- 進行は状況に応じて当日も含め臨機応変の対応をさせていただきます。(Schedule is flexible.)

- 9:00am 開場 (the room open
- 10:00am CELFの現況 (Current status of the CELF)
  - CELFの最近の動きを簡単にお知らせいたします。(Update on CELF, briefely)
  - 今回のイベントの趣旨をご説明いたします。(Explanation of the this event)
  - 参加者の紹介 (Self introduction)
  - 事務連絡そのほか (Administrivia)
- 10:30am ショートプレゼンテーション（午前の部）(Short presentations (Morning session))
  - 10社（グループ）程度を予定 (about 10 companies/groups planned)
- 11:30am 昼食（周囲の混雑を避けるため少し早めの昼食と致します）(Lunch break)
- 1:00pm ショートプレゼンテーション（午後の部）(Short presentations (Afternoon session))
  - 5社（グループ）程度を予定 (about 5 companies/gropus planned)
- 1:30pm ホットトピックスディスカッション (Discussion on hot topics)
  - ショートプレゼンテーションの中から共通の話題をコーディネータがピックアップしてその話題を中心に掘り下げたディスカッション。
    - Common interest from short presentations picked up by coordinators and further detailed discussion
  - 2:45pm コーヒーブレーク (Cofee break)
  - 3:00pm 再開 (Reconvene)
- 3:45pm まとめ (Wrap up)
- 4:00pm 閉会 (Closing)

デモンストレーション / Demonstration

- 昼休みの時間にお願いいたします。会場の後ろ側の座席が使えると思います。当日対応させていただきます。
  - Please demo at lunch break. Seats and tables at the back of the meeting room will be available for demo. Details will be arranged on site.

ジャンボリーの後で / After the Jamboree

- このジャンボリーを通じて更に深く技術検討を進めたい事項が浮き彫りになることを期待しています。その場合CELFは会則の認める範囲の中で下記のご支援を致します。いずれも各メンバー企業では社外の方も含めてアクセス可能なこのようなサイト等の準備は困難かと存じます。是非ご活用いただきますようお願いいたします。手続きの詳細は、追って記載いたします。
  - It is expected through this Jamboree that issues coming up which you want to pursue further technical studies. In that case, CELF would support as follows, within permitted in our bylaws. It seems difficult for each member company to provide such web site like this which even outsiders can access. Please use effectively. Procedures to use will be presented later.
  - その話題に絞ったメイリングリストの設定 (Setting up a mailing list for the specific topic)
    - CELFメンバーは参加可能とさせていただきます
      - Allow CELF members to participate in it

        - CELFメンバーはアーカイブにアクセス可能とさせていただきます
            - Allow CELF members to access the archive
        - 日本語のやりとりでも構いません
            - It is OK to communicate in Japanese.
    - その話題に絞ったWikiページの設定
        - Setting up Wiki pages for the specific topic
        - CELFメンバーはアクセス可能とさせていただきます
            - Allow CELF members to access it
        - CELFメンバー以外にも公開可能です
            - It is OK to make it available for non-members of CELF.
        - 日本語のやりとりでも構いません
            - It is OK to communicate in Japanese
    - 登録メンバー全員に宛てたメイリングリスト、members@list.celinuxforum.orgやCELF外部の開発者も含めたメイリングリスト、celinux-dev@tree.celinuxforum.orgもご活用ください。（メッセージは英文がよろしいかと存じます）。
        - You can use the mailing list for all registered members, (members@celinuxforum.org), or one incluing developers outside of CELF(celinux-dev@tree.celinuxforum.org). Messages in English are desirable.
- 少なくとも一人コーディネータをメンバー企業の中から選んでください。コーディネーターにはメイリングリスト、Wikiページの管理等に関するご連絡をさせていただきます。また、他のメンバーの求めに応じて状況報告をお願いいたします（恐れ入りますが英文にて）。
    - Choose at least one coordinator from member companies. The coordinator will be contacted on administration of mailing lists, Wiki pages etc.. Please report the status (in English, sorry) upon request from other members.
- 正式にワーキンググループを立ち上げる事を目指す事ももちろん可能です。但しワーキンググループ設立までには諸手続が必要で、時間もかかります。まずメイリングリストやWikiを使って技術検討を始められることをお勧めいたします。
    - Formal Working Group can be established, while it requires miscelaneous procedures and takes time. We recommend to start technical study with mailing list and/or wiki.
- また、現在既に活動を開始しているワーキンググループにて検討を進めるという選択肢も有ります。
    - Alternative option might be to participate in an existing Working Group already active.

Meeting Memo and Short Presentation Material

- 宗像さんがミーティング中に書きとめられたメモを、こちらに置きます。/ Here is a meeting memo prepared by Mr. Munakata. Japanese
- 各メンバーのショートプレゼンテーション資料はこちらです。（順不同）。 / Below, you will find the presentation materials of the "Short Presentation" by the attendees.
    - **Coordinator's Note:** Explanation of the current CELF status and the intention of the Jamboree. Japanese
    - **Lineo Solutions:**Japanese
        - Lineo Solutions, Inc., with 19-year experiences in embedded systems development and pioneering experiences in embedded Linux, offers realistic and products focused embedded Linux solutions from our engineering center in Japan where the world dominant development activities in embedded devices are taking places. In upcoming Ubiquitous Age, "Open Standard & Open Source" universality of Linux enables us to expand our uLinux solutions to every corner of embedded systems market in the world.
    - **Sony Corporation:**English
    - **NEC:**Japanese
        - NEC presented the idea to discuss the improvement of the JFFS2 which would be important for the Cellular phone use of Linux. (note by Ueda)
    - **YAMAHA:**Japanese
        - Yamaha is interested in the implementation of Bootloader, implementation of Flash Memory, QoS of audio play back, lower power consumption and MIPS TX49xx overall implementations.
        - They will be able to provide with the Flash-Memory save data and boot up patch for MIPS/TX49xx (zImage/ramdisk/MTD/etc…) and the Boot Loader. (note by Ueda)

- **Mitsubishi:**JapaneseEnglish
- **Fujitsu:**English/Japanese
- **Renesus:**JapaneseEnglish
- **Toshiba:**
  - **Title: NAND Flash Filesystem Improvement** Japanese
  - Abstract: This presentation describes our basic ideas about the optimization of NAND Flash Filesystem for consumer electronics devices.
  - **Title: USB Mass-Storage class Improvement for Removable Media** English
  - Abstract: This presentation describes our basic ideas about the improvement of the functionalities of USB Mass Storage Device Driver for consumer electronics devices.
  - **Title: RBTX49 Board Support** Japanese
  - Abstract: This presentation describes our basic Linux 2.6.X based phase2 development plan about the supporting TOSHIBA RBTX49xx reference board (mips).
- **Hitachi:**Japanese
  - Hitachi summarized the current situation of the Linux development for CE appliances.
  - Then, they pointed out the importance of the further development especially,
    - More reliability
    - More secure operating system, like protecting malicious attack
    - Hot plug in and out of the removable media
    - Device drivers for the low power consumption devidces
    - QoS, like real-time scheduler, isochronous scheduler and control the CPU over load.
  - (summary by Ueda)
- **AXE:**Japanese
  - Axe presented the idea to work on the development of:
    - Improvement of Memory QoS, process priority handling.
    - Handling method of the less memory operation
    - CPU Time QoS.
      - Patch for 2.4 is ready to provide
    - Quick start and supression of memory consumption
      - such as JFFS2 start up time improvement
    - Multi core CPU
  - (summary by Ueda)
- **Sun Wah Linux:**English

Follow-up Meeting

- 第一回東京テクノジャンボリーの結果、更に詳細なディスカッションの場を設定しては如何でしょうか。
  - How about setting up follow-up meeting after the first Tokyo techno Jamboree as the opportunity for further detailed discussion?

ソニー主催のフォローアップミーティング / Follow-up Meeting hosted by Sony

- ソニーは下記の場を設けさせて頂きます。 Japan Technical Jamboree Follow Up (Sony will prepare the following site.)
  - 場所(location)：鶴屋町フォーラム（横浜駅から徒歩10分ほど） (Tsuruya Forum, 10 min walk from Yokohama Station)(鶴屋町フォーラム
  - 日時(date and time)：11月17日（水）午後5時30分〜8時30分(新横浜発21:09のぞみ157号、22:00こだま535号（浜松行）に間に合います。) (November 17, 2004 (wed) 17:30-20:30 (you can go home by Shinkansen even if you live in Hamamatsu.))
- 話題[topics]:
-
  - 1) CPU/Memory QoS (MTA等) / (MTA etc.)

  - 2) USB/Memory Card Hot Plugging (File System関連も含む) / (Including File Systems)
  - 3) Suspend Resume（サスペンド時のファイルシステム保護とリジューム時の高速起動など） / (File system protection at suspending and fast resuming, etc.)
- ご興味を頂けるエンジニアの皆様と突っ込んだ情報交換や、今後の展開に関してディスカッションをしませんか。もしソースコードやスペック等、見せられる物が有ればお互いにレビューしたりできればと考えております。ソニーは1)3)に関してはアウトラインスペックや、完成レベルでは有りませんがソースコードをお見せできます。また、2)に関しては大いに興味があり、是非皆様のお知恵を拝借したく考えております。

  - How about exchanging deep information among engineers with interest, and having discussion on future works? If you have something to show, such as source code or spec, we are expecting that we review them each other. Sony can show outline spec, and unfinished source code, on 1) and 3). We are interested also in 2) and would like to get good ideas from attendees.
- ちょうどパシフィコ横浜でET2004が開催されております。その後にでもお寄りいただけるように、時間と場所を設定させて頂きました。（CELFのブースはA-11です）。
  - We arranged this meeting so that you can drop by after you see the ET2004 at Pacifico Yokohama (CELF booth is A-11).
- 周囲に手頃な飲食店も多い場所ですので、よろしければ終わってから軽くビールでも。・・・これは割り勘で。
  - As there are a lot of restaurants and bars around the meeting site, we can talk more over beers ... please pay your share.

FAQ

Q1:参加者はメンバー企業に限られますか？ / Is an attendee restricted to CELF member company?

- いえ、メンバー企業以外の方もメンバー企業の方のご紹介が有ればご参加頂けます。メンバー企業以外の方がご参加の場合はコーディネーターに一報頂ければ幸いです。(No, non-member can also attend, if invited by member companies. Let us (the coordinators) know if non-member is going to attend.)
- なお、このWikiページはCELFメンバー以外の方もアクセス可能です。(Non-members of CELF can also access this Wiki page.)

Q2:ショートプレゼンテーションは必須ですか？ / Is the short presentation mandatory?

- 必須では有りません。ですが、各位にどのような技術ネタが有るかが解るとジャンボリーも盛り上がるかと存じます。是非一言お願いいたします。口頭のみでもかまいません。もちろん、No obligation / No commitmentで結構です。(Not mandatory, but Jamboree would be successful if attendees see each other what one has technical topics he can talk about. Join us Just oral presentation is OK. Of course, no problem with "No obligation / No commitment".)

Q3:日本語がわからない方も参加できますか？ / Can non-Japanese speaker attend?

- 参加いただいても結構です。但し、翻訳サービス等はございません。(Yes, you can attend. But we can not provide translation service etc..)

Q4:会費はいくらですか？昼食等は用意がありますか？/ How much is the fee? Lunch prepared?

- 会費は無料です。(The fee is free.)
- 昼食は恐れ入りますが用意いたしません。コーヒー等軽い飲み物は用意させていただきます。(No lunch is prepared, sorry. Coffee etc. will be served.)

Q5:デモンストレーションをしても構いませんか？/ Can I demo?

- 商品（ソフトウエアツール、開発ツールなど）のデモンストレーションは昼休みの時間にお願いします。(Please demo products (software tools, development tools) at lunc break.)
- 開発成果物のデモンストレーションに関しては、ご要望になるべくこたえられるようにしたく思います。早めにコーディネーターにご連絡ください。(As to demonstration of your development work, we would like to meet your request as much as possible. Please contact the coordinators as early as possible.)

Q6:次回開催はいつですか？また他の地域でも開催されますか？ / When will be the next event? Will it be held in other regions?

- 今回（第一回CELF テクノジャンボリー・東京）は世界各地域に先駆けてのトライアルとして開催いたします。今のところ次回開催に関しては一切未定です。(This (First CELF Techno Jamboree Tokyo) is held as the trial for the first time in the world. No plan for the next event at the moment.)
- 今回、成功裏に終わりましたら、他地域でも開催される可能性が高くなる筈です。また日本国内で次回の開催も同様です。なお、来年早々に全世界の開発者を対象としたディスカッションの場も計画中ですのでご期待ください。(If this event is successful, the events will likely be held in the other part of the world. So will be the next event in Japan. We are planning the opportunity for the worldwide developers to discuss early next year(2005). Stay tuned)

Q7:イベント開催を待たずに技術検討を始めたい / I can't wait the event and want to start technical study now

- 特定の技術テーマに絞ったメイリングリストやWikiページは、今すぐにでも設定可能です。詳細は英文にてパッチアーカイブメインテナーのTim Bird(tim.bird@am.sony.com)にお問い合わせください。このイベントのコーディネーターに日本語でお問い合わせ頂いても結構です。(We can immediately set up mailing list or Wiki page for specific technical field. Contact Tim Bird, the patch archive maintener, in English. You may also ask the coodinaters of this event in Japanese.)

Categories:

- Events
- Japan Technical Jamboree

> From: eLinux.org

Japan Technical Jamboree 12



Contents

Introduction

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Linux Forum. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- Attendance is not limited to the CELF members, but is open to all. Everyone who is interested in embedded Linux technologies is welcomed. There is no admission fee.
- A general guidance page is available. Japan Technical Jamboree Guidance (Japanese/English)

Coordinators: (Your inquiries in English welcome).

- Hisao Munakata (munakata.hisao(a)renesas.com)
- Satoru Ueda (Satoru.Ueda(a)jp.sony.com)

Table Of Contents

Date and Place

- **12月8日（金） / December 8th (Fri), 2006**
  - From 10:00am to 5pm（会議室は9時まで予約してあります/ Meeting room booked until 9 pm.）
- **Venue**
  - **Nakano Sun Plaza**
  - Homepage (Japanese)
    - English page is not available, however it is very easy to access. Just near by of "Nakano" station, JR Chuo Line and Tokyo Metro (Subway) Tozai Line.

        - The Nakano station is on the Chuo main line, west of Shinjuku station. Here is a Japan Railways map of the Tokyo area
- 場所：中野サンプラザHomepage Location: Nakano Sun Plaza
    - JR中央線・東京メトロ東西線中野駅北口すぐ
- **Good News**: この会場ではインターネットへのフリーアクセスが可能です。当日はWireless LANを設置します。(Free access to the Internet is available. Wireless Lan will be set up on the day.)
- 電源延長ケーブルをお持ちの方はご持参ください。(Come with your AC power extension cable if possible)

Registration

- 参加登録は、こちらのページから。
- Registration site is here.

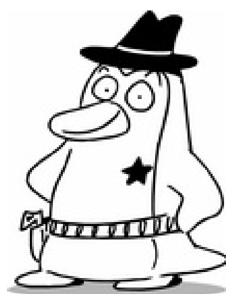
進行の目安 / Agenda

時間 / **Time**	内容 / **Session Title**	資料 / **Materials**
10:00am..10:30	連絡事項・自己紹介*'Opening notice / Self introduction'*	
10:30am..11:30	HTTP-FUSE CLOOP with Software RAID and DNS-Balance for Embedded Linux- Jun Kanai	[1]
11:30am..0:30	Lunch	
0:30pm..3:00	**Secure Operating System Environment**- TBD	
3:00pm..6:00	**Calling for discussion items'**セッションテーマ募集中'	

Discussion Items

- We are now calling for discussion items. Please feel free to send your subject to the coordinators

Special Session



- ***TOMOYO Linux as Secure OS for Consumer Electronics and Embedded Devices*** - Toshiharu Harada (haradats(a)nttdata.co.jp/haradats(a)gmail.com), Tetsuo Handa (hadat(a)pm.nttdata.co.jp)
    - Brief introduction of Secure OS
    - ***TOMOYO Linux*** Concept and Demonstration (on Armadillo-9)
    - ***TOMOYO Linux*** 's advantages for use of Consumer Electronics and Embedded Devices
    - we are also planning public demonstration of ***TOMOYO Linux*** on Armadillo-9, don't forget terminal and LAN cable. (Remember, CELF is no commit B))
    - For your information
        - TOMOYO Linux Homepage (English/Japanese)
            - you can browse the code here
            - Links (Japanese)

- TOMOYO Linux at SourceForge.jp
  - Papers (Japanese/English)
  - Seminar/Symposium (Japanese)
  - Other document stuff (Japanese)
- what is TOMOYO Linux? (Hatena)

-「組み込み用セキュア***OS***としての***TOMOYO Linux***」 - 原田季栄(haradats(a)nttdata.co.jp/haradats(a)gmail.com)、半田哲夫(handat(a)pm.nttdata.co.jp)

-
  - セキュアOSの概要
  - ***TOMOYO Linux***の概要とデモンストレーション (Armadillo-9)
  - ***TOMOYO Linux***の組み込み用途を意識した仕様
  - 上記の他、会場で***TOMOYO Linux*** on Armadillo-9を体験いただけるようにしたいと思っています。端末とケーブルをお忘れなく。（但し、CELFなのであまりあてにしないように B))
  - 参考情報
    - しばらく論文を書いていないので最新の仕様の情報が少ないのですが、***TOMOYO Linux***概要については、Linux Conference 2005の「使いこなせて安全なLinuxを目指して」をご覧下さい。
    - 「難しい話は良いけれど、結局何ができるのか？」という方は、ITproの記事がご参考になると思います（但し、バージョンが古いのでこの手順は実行なさらないように）。
    - 開発の経緯については、Linux Kernel Conference 2005で講演しています。
    - 上記を含めて***TOMOYO Linux***に関して作成した資料は全て公開しており、プロジェクトホームページよりたどれます。
    - 実装にご興味ある方は、技術評論社様のネットワークセキュリティExpert 5に記事を書きました。使い方については、Software Designの1月号より連載が始まりますので是非そちらをご覧下さい。
    - 自分達で動作を確認したのはArmadillo-9ですが、Open Zaurus, Open Block S, グラタン等でも利用できたという報告があります。定期的に巡回して、リンク集に追加しています。

General Session

We are calling for the discussion items

Project Update Session

- **We are calling for the discussion items**

デモンストレーション / Demonstration

- デモンストレーションも歓迎します。
- We welcome your technical demonstration.

はじめて参加される方に(For Newcomers)

-
  - ガイダンスページを作りました。お役に立てば幸いです。Japan Technical Jamboree Guidance (Guidance page has been prepared. Hope it helps.)

お知らせ / Notice

ET 2006

- 今年もCE Linux Forum はET 2006に参加し、ブース設営及びカンファレンスプログラムの中でプレゼンテーションを行います。是非お立ち寄りください。会期は11月15日（火）から18日（金）まで、会場はパシフィコ横浜です。

  - Home Page (English Page available)
  - "ET" (Embedded Technology) is one of major exhibition and conference of embedded system technology which is held in November, Yokohama (Greater Tokyo).
  - CELF Booth and the public presentation will be there,
- ブースでは技術プレゼンテーションを行う予定です。いわばジャンボリーの出前。これによりより多くの人がこのオープンソース開発コミュニティーへ参加するきっかけになることを目論みます。今年、初夏に行われたESEC 2006では狭いながらブースプレゼンテーションを敢行し、多くの方が足をとめてくれました。質疑応答も盛んでした。ETでは少し広めのスペースを確保、本格的に実施します。



- カンファレンスプログラムの中でのプレゼンテーションではフォーラム活動がもたらした技術成果のみならず組込み技術開発に於けるオープンソースの意義、可能性に関してもアピールする予定です。上司の方などにこの活動をよりよく理解していただくきっかけになるように計らいます。

Categories:

- Events
- Japan Technical Jamboree

> From: eLinux.org

Japan Technical Jamboree 2

Table Of Contents

Contents

Introduction

The Japan Technical Jamboree is a forum-wide technical meeting of the CE Linux Forum. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.

このサイトは原則として日本語のみです。 ***THIS PAGE was ;-) JAPANESE LANGUAGE ONLY***

History

The 1st of Japan Technical Jamboree (including Announce, Memos)	Japan Technical Jamboree
The follow up meeting after the 1st Jamboree	Japan Technical Jamboree Follow Up

Latest Update

- 2004/11/16 Registration:Registration has been ready.
- 2004/11/25 Added some information about guests.

A Proposal

- I like to use "san"(NB. a frank postfix in Japanese) in our mail messages. How do you feel it? :)

An Important Notification

- Any future announce/notifications about the Jamborees will be sent not to Members@celinuxforum.org, but ot celinux-dev@tree.celinuxforum.org, which any persons other than CELF members can subscribe for technical information. Those who are interested in future Jamborees are kindly adviced to subscribe it so as to get annouces. Registration here!

The 2nd CELF Techno Jamboree Tokyo

- **Coordinator:**
  - Hisao MUNAKATA
    - munakata.hisao＠renesas.com
    - TEL: 03-3861-2642
  - Satoru UEDA
    - satoru.ueda＠jp.sony.com
    - TEL: 03-6409-5011

Date and Location

- **Friday, December 17, 2004 9:30am to 17:00pm**
  - *: 9:30～10:00 **Tutorial Session Those who attend first are expected to arrive 9:30am,**other people can join 10:00am.
- **Place:**
  - Well-City Tokyo, (or Tokyo Kosei Nenkin Kaikan hall) (Shinjuku-Gyoen Mae)
    - **5 minute walk from SHINJUKU-GYOEN MAE Station, Marunouchi line, Tokyo Metro Railway**
    - **7 minute walk from SHINJUKU-3-CHOME Station, Toei-Shinjuku line**
    - bus from JR Shinjuku Station is also available.
  - Webpage: [1]
  - Map: [2]
  - A room with 140 people capacity is reserved on December 17.
- Some Advices
  - Not enough number of 100volt power outlets will be available. Those with power codes are welcomed.
  - No internet connectivity is provided. PHS wireless service will be available at the window-side of the hall.
- **Dress code is Kajual(casual) (means "no tie" in Japanese)**

Registration

- こちらのページで参加登録願います。この登録ページはCELFメンバー登録されていない方はアクセス出来ません。もしCELFメンバー登録をされていない方が登録ページから参加登録される場合は、(Register for the Jamboree at this page. Only registered members of CELF can access this registration page. Otherwise:)
    - どなたかメンバー登録されている方に登録していただく(Ask somebody who is already registerd as a CELF member.)
    - または、ご自身でメンバー登録していただく(Or, register yourself as CELF member here.)
    - または、コーディネータにご連絡願います。コーディネータが登録させていただきます。(Or, contact the coodinators. The coordinator will register on behalf of you.)

ショートプレゼンテーション / Short Presentation

- 今回初めて参加される方は、ショートプレゼンテーションをいただければ幸いです。各社**5〜10**分程度を目安。もし各社の中で異なる技術開発項目を持たれている複数のグループが参加される場合は、それぞれグループ毎にプレゼンテーションしていただいても結構です。下記のいずれかの内容でお願いいたします。
    -
        - We appreciate the newcomers make a short presentation, 5-10 minutes for each. If more than one groups with different technical, development issues in one company, each group may make a presentation. Choose one from below:
- また前回のショートプレゼンテーションに追加等がある方もショートプレゼンテーション頂ければと思います。
    - The additional presentation, if any, from the repeaters will be welcome.
    - どのような技術分野に関してどのようなパッチをご提供いただけるか
        -
            - what patches you can submit in what technical field?
    - どのような技術分野のいかなる事で開発に難儀しているか
        -
            - what issue you are struggling with?
    - どのような事に関してお互いに情報交換したいか
        -
            - on what topics you would like to exchange information each other?
- ご承知の通り、CELFはGPL/LGPLないしはそれらに互換性があるライセンスによって自由に交換できるソフトウエアをベースに技術ディスカッションを進める場です。上記に関しましてもその条件を前提としてお話しいただければと思います。
    -
        -
            - As you know, CELF is a place for technical discussion based on software freely exchangeable with GPL/LGPL or compatible licenses. Please talk with that assumption.
- このプレゼンテーションは必須ではありません。またプレゼンテーションの内容に関して一切のお約束等は無くて結構です。No commitment, No obligationで結構です。
    -
        -
            - This presentation is not mandatory. And you do not have to commit anything on the content of your presentation. No commitment/No obligation.
- プロジェクターは用意いたします。恐れ入りますがプレゼンテーション用のPCは各位でご準備ください。
    -
        -
            - We prepare a projector. Please come with your own PC for presentation.

Agenda（仮） (tentative)

- 進行は状況に応じて当日も含め臨機応変の対応をさせていただきます。(Schedule is flexible.)
    - **8:30am** Hall Opening
    - **9:30am** Tutorial Session
        - A brief report of recent CELF activities.
        - Purpose/Concept of this Jamboree
    - **10:00am** Start Session
        - Self introduction
        - Announcements
    - **10:10am** Short Presentations
        - Five Companies (groups)
    - **10:45am** Main Session
    - **11:30am** Lunch (jump start to avoid official office lunchtime).
    - **1:00pm** Main Session (continue)
    - **3:00pm** Coffee Break
    - **3:15pm** Main Session (continue)
    - **4:30pm** Summary Session
    - **5:00pm** Adjorning

Demonstrations

- Demos will be welcomed in lunch time. 会場の後ろ側の座席が使えると思います。当日対応させていただきます。(Seats and tables at the back of the meeting room will be available for demo. Details will be arranged on site.)
- 前回と異なり今回は場所、時間に余裕が有ります。デモンストレーションご希望の方で事前に会場に機材を別送したい方は、上田宛お問い合わせください。会場にあらかじめ連絡させていただくと同時に送り先をお伝えいたします。(Unlike the previous event, we have a plenty of space and time. If you are going to demo and want to ship the equipments beforeand, please contact Mr. Ueda. He will notify to the site and let you know the shipping address.)

メインセッション / Main Session

\<> ご注目願います！ Beware

- 今回は、前回のミーティングのディスカッションを更に深めます。前回各社の提供可能な技術要素や、開発にチャレンジしたいと思っている事が浮き出してきたかと思います。
    -
        -
            - In this event, we would like to deepen the discussion at the previous meeting. At the last meeting, technical components each attendee can contribute and/or is going to challenge have shown up.
- それらをふまえて今回更に深い技術討論の場を持つ事とします。今回は皆様からサンプルコードや技術概要がわかる資料をお持ち頂ければと存じます。前回のミーティングに関しては、 Japan Technical Jamboree を参照してください。
    -
        -
            - Based on them, we are going to have deeper technical discussion. This time, we would like you to attend with your sample code or documents which present technical issues. For the previous meeting, see Japan Technical Jamboree.
- また、事前にどのような技術テーマについてディスカッションしたいかを皆様にお伝えできればそれぞれ事前準備も可能かと思います。採り上げたいテーマを是非コーディネータにあらかじめお伝えください。Wikiページのこのセクションに記載させていただきます。直接このセクションを編集していただいても結構です。
    -

-
    - If attendees know beforehand what technical issue you want to discuss, each attendee would prepare for it. Let the coordinators know what topic you want to pick up. The coordinators will write them on this section of Wiki. Or you can directly edit this section.

CPU/Memory QoS （Sony）

- A detailed study for some technologies, including MTA.
- Plan to show some sample source codes.

USB/Memory Card Hot Plugging （Sony）

- A detailed study for some technologies, including File System features.

Suspend and Resume （Sony）

- A study about file system protection in suspending and about quickboot in resuming.
- Plan to show some sample source codes.

USB Media Hotplug Detectio (Toshiba)

- Plan to show some sample source codes.

An improved imprement for media detaching during accessing the media (Toshiba)

- Plan to show some sample source codes.

A study how to register unusual device list (Toshiba)

DirectFB 及び 3D Graphics （ルネサス提案） / (Renesus proposal)

- DirectFB の実装状況説明（最新ベンチマークデータの公開など）
    -
        -
            - Implementation status of DirectFB (The latest benchmark result, etc.)
- 組み込み Linux における 3D Graphics への要求などの審議
    -
        -
            - Discussion on requirements for 3D Graphics in embedded Linux.
- DirectFB による 2D／3D API の統合の可能性審議
    -
        -
            - Discussion on the possibility to integrate 2D/3D API with DirectFB
- このセッションで議論した内容を DirectFB 開発者等に Feedback しようと考えています
    -
        -
            - The result of discussion in this session will be feedbacked to the DirectFB developers etc..

ゲスト / Guests

- 今回下記のゲストが参加します。(Following guests will join this event.)

経済産業省・商務情報政策局・情報処理振興課、久米課長補佐 / Mr. Takashi Kume, Deputy Manager, Commerse and Information Policy Bureau, Ministory of Economy, Trade and Industry(METI)

- 政策立案に活かすため、組み込みLinuxの最前線の実状を把握したいとの事です。(He wants to understand actual circumstances of the embedded Linux for policy-making.)
- ジャンボリーの中で15分程度、経済産業省に現場の声を届けるセッションを設けたく思います。是非、忌憚無き声を！ (We plan to have 15 minute session to deliver our voices from the hot scene to METI. Talk loud please)
- なお、経済産業省商務情報政策局はBlogサイトを立ち上げています。久米さんのサイトも有ります。ご存知の方も多いかと思いますが、情報家電関係のソフトウエア基盤整備に係わる政策立案に情熱を傾けられている方です。(Commerse and Information Policy Bureau of METI have a Blog site \ which Mr. Kume coauthored. As you may know, he is eager in policy-making on software infrastructure for information appliances.)

Emblix 中島会長（早稲田大学教授）/ Prof. Tatsuo Nakajima, Waseda Univ.; EMBLIX Chair

- Short Presentationで、特に"CPU QoS"に関して等、Emblixでの取り組みと成果を披露していただきます。(As short presentation, he talks about their approach and result at EMBLIX, especially on "CPU QoS".)
- その後のディスカッションにも参加されます。(He will join the further discussion.)

日経エレクトロニクス / Nikkei Electronics magazine

- 日経エレクトロニクス記者の参加(Reporter of Nikkei electronics magazine)
  - 記事にする場合は、ジャンボリーの中での発言等が誰の発言か特定できるような掲載はしないようにお願いしました。これは**"3N"**の原則に対する配慮です。(We have asked him not to report in such way that readers of the articles can find who is the speaker, according to our 3N Rules.)

CQ出版社 / CQ Publishing Co. Ltd.

- インターフェース誌編集長(Editor in Chief, Interface magazine)
  - 記事にする場合は、ジャンボリーの中での発言等が誰の発言か特定できるような掲載はしないようにお願いしました。これは**"3N"**の原則に対する配慮です。(We have asked him not to report in such way that readers of the articles can find who is the speaker, according to our 3N Rules.)

Meeting Memo

- 宗像がミーティング中に書きとめられたメモを、こちらに置きます。/ Here is a meeting memo prepared by Mr. Munakata. Japanese Japanese(pukiwiki format)

Presentation Material

- 各メンバーによるプレゼンテーション資料はこちらです。（順不同）。/ Below, you will find the presentation materials by the attendees.
  - **Renesas _3D graphics:**JapaneseEnglish
-
  - **Fujitsu privileged interrupt facility:**Japanese
-
  - **Ricoh (Mr. Alan Volmat) presented "Linux on a Digital Camera"** Media:ricoh_jamboree2.pdf (English)

Follow Up Meetings

- May 20, 2005: Discussion about the XvFAT implementation.
  - Hosted by Sony
  - Details : Xv Fat Discussion

FAQ

このジャンボリーに何か特別なルールは有りますか? / Is there any special rule for this Jamboree?

- **3N** ルールが有ります。3Nとは・・・、(We have "3N" rules, where 3N means:)
  - No Confidentiality: 守秘義務は一切有りません。
  - No Obligation: 義務や責任は一切有りません。
  - No Commitment: 約束無しで構いません。
- CELFはオープンコミュニティーです。"**Open Source Community Way**"で！ :) (CELF is an open community. We do "Open Source Community Way" :))

参加者はメンバー企業に限られますか？ / Is an attendee restricted to CELF member company?

- いえ、メンバー企業以外の方もメンバー企業の方のご紹介が有ればご参加頂けます。メンバー企業以外の方がご参加の場合はコーディネーターに一報頂ければ幸いです。(No, non-member can also attend, if invited by member companies. Let us (the coordinators) know if non-member is going to attend.)
- なお、このWikiページはCELFメンバー以外の方もアクセス可能です。(Non-members of CELF can also access this Wiki page.)

ショートプレゼンテーションは必須ですか？ / Is the short presentation mandatory?

- 必須では有りません。ですが、どのような技術ネタが有るかが解るとジャンボリーも盛り上がるかと存じます。特に今回初めて参加される方は是非一言お願いいたします。(Not mandatory, but Jamboree would be successful if attendees see each other what one has technical topics he can talk about. Especially we are expecting presentation from newcomers".)
  - 口頭のみでもかまいません。**5〜10**分程度でお願いします。(Just oral presentation is OK. Give presentation in 5-10 minutes.)
  - もちろん、No obligation / No commitmentで結構です。(Of course, no problem with "No obligation / No commitment".)

日本語がわからない方も参加できますか？ / Can non-Japanese speaker attend?

- 参加いただいても結構です。但し、翻訳サービス等はございません。(Yes, you can attend. But we can not provide translation service etc..)

会費はいくらですか？昼食等は用意がありますか？ディナーパーティーは？ / How much is the fee? Lunch prepared? Dinner Party?

- 会費は無料です。(The fee is free.)
- 昼食は恐れ入りますが用意いたしません。コーヒー等軽い飲み物は用意させていただきます。(No lunch is prepared, sorry. Coffee etc. will be served.)
- ディナーパーティーの準備も有りませんが、時節柄、また新宿御苑界隈という場所柄もあります。有志で軽く・・・？ (No plan of the dinner party. But it's Christmas season, and the place is near Shinjuku Palace. We will be able to have a chat over drink...?)

デモンストレーションをしても構いませんか？ / Can I demo?

- 商品（ソフトウエアツール、開発ツールなど）のデモンストレーションは昼休みの時間にお願いします。(Please demo products (software tools, development tools) at lunch break.)
- 開発成果物のデモンストレーションに関しては、ご要望になるべくこたえられるようにしたく思います。早めにコーディネーターにご連絡ください。(As to demonstration of your development work, we would like to meet your request as much as possible. Please contact the coordinators as early as possible.)
- 今回は場所、時間に余裕がありますのでデモンストレーション歓迎です。(In this event, we have a lot of space and time for demo. Any demos are welcome.)

他の地域でも開催されますか？ / Will the event be held in the other region?

- 他地域でも開催される可能性が高くなってまいりました。詳細は未定です。(Jamborees at other regions are getting likely. TBD.)
- 来年1月に全世界の開発者を対象としたカンファレンス（CELF総会）が有ります。詳細はここをご覧ください。この総会は**CELF**メンバー（メンバー企業従業員）は誰でも参加できます。(A conference for worldwide developers (and CELF plenary meeting) is being held in January next year (2005). See here for details. Any members of CELF (emploees of member companies) can attend this plenary meeting.)

イベント開催を待たずに技術検討を始めたい / I can't wait the event and want to start technical study now

- 特定の技術テーマに絞ったメイリングリストやWikiページは、今すぐにでも設定可能です。詳細は英文にてパッチアーカイブメインテナーのTim Bird(tim.bird@am.sony.com)にお問い合わせください。このイベントのコーディネーターに日本語でお問い合わせ頂いても結構です。(We can immediately set up mailing list or Wiki page for specific technical field. Contact Tim Bird, the patch archive maintener, in English. You may also ask the coodinaters of this event in Japanese.)

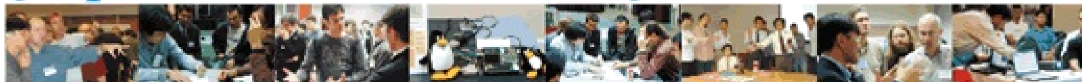
Categories:

- Events
- Japan Technical Jamboree

> From: eLinux.org

Japan Technical Jamboree 27

CE Linux Forum
Japan Technical Jamboree



Date: May 22nd / 日付: 5月22日（金）
At Nakano Sunplaza / 於、中野サンプラザ

Contents

Shorten the time / 時間短縮のお知らせ

- Please be noted that we have shortened this event duration time because of suddern expansion of swine flu epidemic situation in Japan.
  - **We will start the sessions at *1pm***.
- 新型インフルエンザの日本における急速な流行拡大の兆しを受けて、開始時間を午後*1*時とし、開催時間を短縮して開催します。
  - 参加に際しましてはマスクの着用や手洗いの励行を心がけてください。
  - 今回は遅くとも5時頃には終えるようにします。
  - 各セッションはビデオ撮影し、今回参加できなかった方にもご覧頂く機会を設けます。

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Linux Forum. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
  - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly without any complicated expressions. Most Japanese developers are capable to understand plain English.

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。
    - http://tree.celinuxforum.org/CelfPubWiki/JapanTechnicalJamboree26 (In the CE Linux Forum Public Wiki)

Date and venue... / 日付・場所...



- Date **May 22, 2009**
    - ***Starting at 1 pm***
- At ***Nakano Sunplaza*** / 会場 中野サンプラザ
    - Nakano Sunplaza is located just close to **Nakano** station (JR/Tokyo Metoro).
    - http://www.sunplaza.jp/ (Japanese)
- Admission: ***Free of charge*** / 参加費用: 無料
- Not limited for CELF members. / CELF会員以外も参加・セッション持ち込み共に可能
- Coordinators / 世話役 (Your inquiries in English welcome)
    - Hisao Munakata / 宗像尚郎 (munakata_dot_hisao(a)renesas_dot_com)
    - Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

Registration / 参加登録

- Registration Page / 参加登録
    - Registration is only to forecast the approximate number of the attendees. No confirmation mail will be delivered. It is not mandatory.
    - 参加登録は大まかな参加人数把握のためにお願いしています。入力をされても確認メールは発信されません。参加登録をしないで出席していただいても構いません。

Main Topics

- Coming back from **Embedded Linux Conference 2009** and **Collaboration Summit**.

ELC 2009とCollaboration Summitの帰朝報告

Agenda / 進行

Agenda

Time	Title and presentor	Notes Presentation Materials
1:00pm..	TomoyoLinux on Android (By Daisuke Numaguchi, Giuseppe La Tona, Tetsuo Handa/ NTT data)	Part1 Outline (Japanese) Video Part2 TOMOYO on Android (English) Video
2:00pm..	Collaboration Summit Report about Moblin (By M. Amano / Miracle Linux)	Media:CELF_Japan_Technical_Jamboree_20090522.pdf (mostly English) Video
3:00pm..	組込みLinux開発環境へのATA over Ethernet適用 (By T. Shigeoka / Hitachi)	Media:AoEForEmbeddedLinuxDevEnv-CELF200905shigeoka-jp.pdf(Japanese) AoEForEmbeddedLinuxDevEnvEngText(english/text) Video

- Please be noted above time table is just a gudeline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWikiに残してください。

English Translation Volunteer

- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

Categories:

- Japan Technical Jamboree
- Events

> From: eLinux.org

Japan Technical Jamboree 28



Date: June 12th / 日付: 6月12日（金）
At Shin-Osaka Station Hotel Annex / 於、新大阪ステーションホテルアネックス

Contents

Special Notice / 注意

- The epidemic condition of swine flu in west Japan area is calming. This event will be held as scheduled.
- 関西に於ける新型インフルエンザの流行が沈静化してきました。このイベントは、予定通り開催出来る見込みです。

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Linux Forum. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
  - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly

without any complicated expressions. Most Japanese developers are capable to understand plain English.

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。
    - Japan Technical Jamboree 27

Date and venue... / 日付・場所...

<table>
<tr>
<td></td>
<td>

- Date **June 12, 2009**
    - Starting at 10 am
- At ***Shin-Osaka Station Hotel Annex*** / 会場 新大阪ステーションホテルアネックス
    - Shin-Osaka Station Hotel Annex is located just close to **Shi-Osaka** station (JR/Osaka Metoro).
    - http://www.st-hotel.jp/annex/access/index.html (Japanese, Page in English available)
- Admission: ***Free of charge*** / 参加費用: 無料
- Not limited for CELF members. / CELF会員以外も参加・セッション持ち込み共に可能
- Coordinators / 世話役 (Your inquiries in English welcome)
    - Hisao Munakata / 宗像尚郎 (munakata_dot_hisao(a)renesas_dot_com)
    - Shinsuke Kato / 加藤慎介 (kato_dot_shinsuke(a)jp_dot_panasonic_dot_com)
    - Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

</td>
</tr>
</table>

Registration / 参加登録

- Registration Page / 参加登録
    - Registration is only to forecast the approximate number of the attendees. No confirmation mail will be delivered. It is not mandatory.
    - 参加登録は大まかな参加人数把握のためにお願いしています。入力をされても確認メールは発信されません。参加登録をしないで出席していただいても構いません。

Main Topics

Agenda / 進行

Agenda

Time	Title and presentor	Notes Presentation Materials
10:00am..	• Opening and Selfintroduction / 連絡事項・自己紹介	
10:30am..	• Collaboration Summit Report about Moblin (By M. Amano / Miracle Linux)	**Video Session** • Video ◦ Video was taken in previous Jamboree (#27) • Pdf File (mostly in English)
11:30am..	Lunch break	
12:30pm..	• TomoyoLinux on Android (with demo) (By Daisuke Numaguchi, Giuseppe La Tona/ NTT DATA)	• Video 1(Special remarks from Harada san.) • Part1 Outline (Japanese) ◦ Video 2 • Part2 TOMOYO on Android (English) ◦ Video 3
13:30pm..	• Git howto (Nobuhiro Iwamatsu/Renesas)	• Video • presentation file
14:30pm..	• Load to add new architecture in Debian (Nobuhiro Iwamatsu/Renesas)	• Video • presentation file
15:00pm..	(Break)	
15:30pm..	• Latest Kernel Development Status Watch (Kosaki)	• Pdf File ◦ (with English annotation) • Video
16:30pm..	Lightening Session / 飛び入りの時間です	
17:30pm..	(we are calling for your session proposals / セッション提案募集中)	Japan TJ Session Proposal

- Please be noted above time table is just a gudeline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

Meeting Memo

- Meeting Memo by "Maccya-Daifuku" Memo (Japanese)
- Followings are donated by Fujimura san with many thanks!

```
--------------------------------------------------------------
ジャンボリーについて

上田さん / Sony
--------------------------------------------------------------
■ ジャンボリーについて
```

```
・CELFジャンボリーの活動は、海外でも注目されてきた。
 →レポートが英訳さてていて、メインラインの技術者も見ている。

・今回のWebサイト
 http://elinux.org/Japan_Technical_Jamboree_28

・次回は7月17日
 中野サンプラザ

・Japan Linux Symposium開催（10/21～23に東京）
 （The Linux Foundation主催）
 →「今年は重要な年」になる位置づけ
 前日に、「カーネルサミット」がある。
 カーネル技術者がハイレベルでディスカッションする場。
 カーネルサミットの翌日なので、海外から来たエンジニアの
 ほとんどが集まるはず。

 10/21のキーノートは、Linusが話をする。
 翌日からのテクニカルセッションにも、組み込み向けのセッションが
 多数あります。

 8/1までに登録すると、$200（それ以降は$300）
   (8/31とお伝えしましたが8/1の誤りでした。)

-----------------------------------------------------------
Collaboration Summit Report about Moblin

天野さん / Miracle Linux
※Skypeで東京から講演。
-----------------------------------------------------------

■ Moblinとは
・Atomベースに最適化されたディストリビューション
・Intelから、コミュニティベースの開発に完全移行された。

■ 天野さん
・ミラクル・リナックス
・moblinプロジェクトには、1年ぐらい前から参加
・開発ツールのメンテナ

■ Moblin概要
・Linux Foundation Collaboration Summit（LFCS）から
 ピックアップして紹介
・モバイル向けLinuxプラットフォームを推進
・フォーカスしているのはケータイ（MID）
 車載、組み込みボードをターゲットにしている。
・プロジェクトの歴史
 2007年7月にプロジェクト公開
・moblin v1
 ubuntu（dev）のパッケージ管理
   →パッケージコンポーネントとして提供
・moblin v2
 RPMパッケージ管理
   →ディストリビューションとして提供
・2009年1月に、Intelが、LinuxFoundationに移管した。

■ moblin v1
・GNOME mobileベース
・カーネル2.6.24
 →チップセット用のパッチを提供
・Xserver1.4
・Ubuntu8.04ベース

■ moblin v2
・ディストリビューション提供を考え、ソフトウェアモジュールを提供
 -3Dレイヤー
   OpenGLベースのライブラリ
・カーネル2.6.29
 -CONFIG_FASTBOOT=y
・Xserver1.6

■ CONFIG_FASTBOOT
```

・SSD最適化（先読み機能（sreadahead）を最適化）
・カーネルカスタマイズ
  -デバイスモジュールは組み込み。（ロード時間の短縮）
  -ネットブック用や、MIDなどの提供形態
  -非同期・並列初期化による高速化
・initrdは使っていない。boot splash無効
・sysvinit
・グラフィックドライバxorg-x11-drv-intelの調整
・起動時間はコールドブートで10秒ぐらい、ベータでは6-7秒になった。
  →目標は2秒
    ただし、ブートローダーからの時間（BIOSは除外）

■ 開発体制
・Bugzilla，Git，メーリングリスト，IRC
・ベータがリリースされてから、開発が活発になってきた。

■ Pickups from LFCS
・LFCSはホテルで開催。
  複数のセッションで、技術発表が行われた。（Intel中心）
・librest，mojito
  twitterやFlickerなどのソーシャルネットワークと、GUIを統合するAPI
  →つなげるアプリを簡単に作れる
・Connection Manager
  -モバイルシステム用のネットワークマネージャ
  -過去のネットワークマネージャは、様々な機能のため拡張が難しかった。
    バグはディストリビューターがPatchしていた。
    GNOMEに依存した実装など、問題があった。
  -このため、moblin用のネットワークマネージャを独自で作った。
  -有線LAN，無線LANなどをプラグイン化。
  -デーモンとして動くとメモリを食う。
    必要最低限のリソースだけ使うように、デバイスごとにプラグイン化
  -即時接続
  -DNSはローカルホストを参照するなどの工夫

・Clutter Project
  -OpenGLのライブラリを簡単に使えるように提供する。
  -3Dユーザーインターフェースライブラリ
  -ほかのGUIライブラリ、Webサービスと統合して実装できる。
    Qtライブラリ、GTKライブラリ、
    Webブラウザ
    物理エンジン
    これらを、clutterから統合して使える。
    Cでも作れるが、Python，C#などからbindして使える。

・Moblin SDK
  -KVM，VMWare実行イメージ
    USBブートイメージなどで提供
  -ツール
    イメージ生成ツール
    LinuxProjectGeneratior（Autotoolsなどを簡略化して使える）
  -ドキュメント
    APIリファレンス，DevelopersGuide

■ 将来の展開
  -現在はネットブック向けに提供を予定しているが（09/Q3）、
    来年あたりにMID向けにも提供
  -次の世代のネットブック向け

■ 日本での情報
・日経Linux2008/09，2009年7月から連載
・ITProの記事

■ Q&A
・Atomアーキテクチャのサポートしているが、32bitだけか？
  AtomはCore2と同じアーキテクチャを採用しているので、
  64bitも対応できるだろう。現状は、32bitのみ提供。
・debからrpmに移行した理由は？
  ちゃんとした答えが返ってきてない。
  もともとのrpm，debの特徴もあるが、開発者がrpmが使いやすいと
  判断したらしい。
・LinuxProjectGeneratior
  -組み込みは、機種展開、アーキテクチャのスケーラビリティがかぎになる。

```
  -コンフィグレーションの自由度（アーキテクチャ対応）や、
    ビルドシステムのスケーラビリティ
    そして、イメージ生成ツールとの連携などは工夫されているか。
・Anacondaでカスタマイズ
  カーネルのコンフィグレーションはパッケージを組み込んだり、
  アーキテクチャごとのext3ループバックイメージを提供
・LinuxFoundationで、moblinのみのイベントも企画している。
  もう少し深いお話もできると思います。

------------------------------------------------------------
TOMOYO Linux / TOMOYO Linux on Android

沼口さん / NTTデータ
Giuseppe La Tonaさん / NTTデータ
------------------------------------------------------------

■ 最初に
・2.6.30リリース
  -TOMOYOのリーダー原田さんよりご挨拶。
  -TOMOYO Linuxがリリースされた。
  -日本で始めて、メインラインをやったら？といわれたのは、
   CELFジャンボリーが最初。
    http://sourceforge.jp/projects/tomoyo/wiki/ThankYou
  -終わりではなく、スタート。
   メインラインを保っていく。
   事業として確立する。
  -TOMOYOは、Linux進化の第一歩に入った。
   これからは、原田さんだけでなく、使う人たちが発展させていかなければならない。
   組み込みでも、この発展に寄与できるものは、どんどんやっていきたい。

■ TOMOYO Linux（沼口さん / NTTデータ）

■ TOMOYO Linuxの概要
・「つかいこなせて安全」なLinuxを
・カーネルパッチとユーティリティの２つから構成
・ディストリビューションではない
・メインライン版
・フル機能版
  →LSMを使ってない、独自フックのもの
・RedHatなどのディストリビューションで開発が進む

・特徴
  -「パス名ベース」で、ポリシー編集が簡単
  -システムの起動から終了まで、「自動学習」
    自分のシステムの把握にも便利。

・導入のメリット
  -被害の局所化
    被害を限定できる
  -不正アクセスの検知
  -誤操作防止
  -情報漏えいの可能性を軽減

・組み込み機器とTOMOYO
  -自動学習
    テストケースがきまっていれば、動作させただけで、ポリシーを反映できる。
  -リンクの区別
    シンボリックリンク，ハードリンクの区別
  -ファイルシステムの制限が無い
  -省メモリ
    ４MBで動作
  -2.4カーネルでも使える

・SE Linuxとの比較
  -いろいろと違いがあるが、棲み分けができるもののはず

・たとえば、使い方
  -Webサーバー
    読み込み専用にしたりできる
  -アプリケーション
    操作を限定できる。
  -ファイル
```

```
    読み書き実行制御，名前空間制御
  -ネットワーク
    IPアドレスなど

・利用の流れ
  -カーネル、ユーティリティをインストール
    ↓
  -システム動作を学習
    ↓
  -自動作成されたポリシーを確認・修正
    学習、修正を何回かまわす
    ↓
  -運用

・TOMOYO のポリシー
  ドメイン＋制御モード＋アクセス許可

  -アクセス許可
    ホワイトリスト方式
  -ドメイン
    すべてのプロセスはいずれか1個のドメインに所属
  -制御モード

・デモ
  SSHの動作制御
  学習させたコマンドだけ実行できる

・Q&A
  -いちから動作をすべて網羅させるのは大変では？
    →現状、システムテスト段階でポリシーを自動生成させるのが一番一般的な方法
      あるサーバーシステムでは、2週間、テスト状態で放置し、すべてのプロセス、
      CGIを網羅させる。
  -ハードが壊れたときのフェイルセーフの処理が走らないなどが懸念されるが、
    回避策はありますか？
    →2002年にやった導入事例では、フェイルセーフのプロセスを走るようにして、
      対応していた。
      ただし、ハードの故障時の対応などは対応が難しいところもある。
    →ターボLinuxは最初にポリシーファイルが配られている。
  -ポリシーのノウハウがたまれば、配布も考えている。
    現在も、wikiでポリシー募集しているが、いまはユーザーが少なく、
    うまくまわっていない。
    →「こういうポリシーつくったよ！」があれば、教えてほしい。


■ TOMOYO Linux on Android（Giuseppe La Tonaさん / NTTデータ）

  -インターンでNTTデータの研究所で勉強中。
  -セキュアOSグループでTOMOYOを研究中。

・Android概要
  -AndroidSDK1.5では、Linux2.6.27をサポート。
    2.6.29も準備している

・Androidの起動プロセス
  -アプリケーションは、dalvikVMで動くJavaアプリケーションである。
  -ZygoteProcess
    zygoteProcessが、DalvikVMをforkし、アプリケーションが動く。
    zygoteがリソースを管理する、idleプロセスである。

・Androidのセキュリティモデル
  -すべてのプロセスは、DalvikVMの区切られたプロセスである。
  -すべてのプロセスは、「secure sandbox」である。
  -LinuxのDAC（アクセスコントロール）はall
  -アプリケーションには、ユニークなUIDが割り当てられる。
  -アプリケーションのデータは、/data/dataに、UIDのフォルダで区切られて管理される。

・AndroidのTOMOYOパッチ
  TOMOYOLinuxによるポリシーを用いて接続制御を行う
  TOMOYO Linux1.6.8(non-LSM)を適用
  Goldfishエミュレータ（ARM）用のクロスコンパイル
  ccspatch1.6.8にパッチ
    →ccstoolsは、組み込み向けに、サイズ削減、ポリシーエディタの簡略化で対応
```

```
  Androidには、ポリシーローダーとして、/bin/css-toolをインストール
  /data/css
  /system/css
    でセキュアなデータ，ファームウェアを管理

・Androidのポリシーファイル
  UIDごとに異なるアプリケーションのドメイン制御が問題だった。
  すべてが、app_processになるため、アプリケーションごとのドメイン制御が難しい
    解決策として、インターネットアプリケーションにくるタスクのUIDによって、
    アクセス制御できるようにした。
  （例）
    allow_read/write @APP_DATA_FILE if task.uid=10001
    allow_read       @APP_DATA_FILE if task.uid=10001-10002

・デモ
  -ポリシーデーモンが動いている状態で、operaブラウザを動作させて、
    ポリシーを覚えさせる。
    →ほかのアプリケーションがネットワークにつなごうとしても、
      ブロックされることを確認。

  -toolboxのシンボリックリンクも、ポリシーエディタが
    ひとつのプロセスとして認識してくれる。


・動作環境
  -カーネルメモリは43kbぐらいの増加
  -ポリシーデータで80kb程度のハンドリング


・Q&A
  後でインストールしたアプリが使うデータも、ワイルドカード指定などで、
  編集可能な領域を指定できる。
  また、ネットワークアプリケーションのuid許可も、パス名で安全性の制御ができる。

  インテントの制御はできるか？
    →ioctlの制御が複雑で、TOMOYOで追っかけるのは難しかった。
      ただ、一通り動かせば、アプリの動作は追えるはずなので、
      それでポリシー制御をしてみては。

------------------------------------------------------------
Linuxカーネル開発者予備軍のためのGitHowto

岩松さん / ルネサス
------------------------------------------------------------
■
どうやって、Gitにパッチを送るのかをテーマにした。


■ 岩松さん
・SHのカーネル開発
・SHブートローダーのメンテナ
・debianのメンテナ


■ Gitの基本的なコマンド
・Subversion（集中管理型）との違い
  リポジトリにコミットするには、コミット権が必要
  ワーキングコピーを作る
・コミット、チェックアウトはローカルリポジトリへの操作
・プッシュ、フェッチはリモートリポジトリへの操作

・Git
  作業者もリポジトリを持つ。
  ローカルでコミットして、リモートリポジトリにマージする。

・基本コマンド
  14個（多い！）を覚えれば、すべて使える！

  ワーキングコピー、index、ローカルリポジトリ、リモートリポジトリ。
  4つのステージを理解すれば、動作の流れをつかめる。
```

■　Gitを使った開発の流れ

・流れ
　-パッチは、開発者の最新のコミットに対して修正を送る
　-カーネルでは、各機能ごとにメンテナがいるので、
　　メンテナンスしてるリポジトリに対して修正を行う。
　-メーリングリストにも送る。

・環境の設定
　-名前とメールアドレス
　　「git config」で設定。システム全体に反映させる
　-リポジトリをコピーする
　　SHアーキテクチャのGitリポジトリはMAINTAINERSファイルに書いてある。
　　「git clone」でコピーする
　-ローカルリポジトリではorigin/headsという扱いでリモートリポジトリの
　　ブランチを管理する　※originはリモートリポジトリの状態
　-cdでソースディレクトリに入る
　-不具合の修正をする
　　修正前と修正後のdiffをみる
　-修正ができたら、コンパイル、テストをする
　-コミットする
　　「git commit」
　　-sオプションは、Signed-offをつけるという意味
　　　エディタが立ち上がるので、コメントを書く。
　　　Signed-offをつけると、名前とメールアドレスが入る。

　　カーネルの場合は、メールのサブジェクトになる。
　　空行を入れて、次の行からがメールの本文になる。

　-git logでメッセージと差分を確認。
　「-p」でコミットしたときの差分とメッセージも表示してくれる
　　→これで、ローカルのブランチ（heads/master）が変更される。

　-修正している間に、リポジトリが更新されている可能性があるので、
　　状態をアップデートする。
　　「git remote update」

　　　origin/masterブランチが変更される。

　　　→ここで、リモートリポジトリの状態に対して、rebaseを行う。
　　　　「git rebase origin」

　　　　mergeは、共通のコミットを探して、それに対して差分適応するので、
　　　　注意が必要。（普通は使わない。）

　-メールに送るpatchは
　　「git format-patch」で生成できる。
　　　→メールのSubject，本文まで生成してくれる。

　-パッチをメールで送る
　　「git send-email」でOK。

■　最新機能を試す

　-最新のドライバとか、最新の機能とか
　-試す必要があるというか、試す必要がある場合がある

（例）sh-2.6のリポジトリをベースにしたブランチにbluetoothの新機能を取り込む

・MAINTAINERSにgitリポジトリが書かれている。
・git remoteコマンドで、リモートリポジトリを追加する。
　git remote update bluetoothで最新の状態にアップデート
　git branchでmasterブランチを作成
　git mergeで、blutooth/masterの状態を反映

　git show headをすると、マージ情報も取得できる。

　　→コンパイル、テストをする。

```
■ まとめ
・リモートリポジトリと、ローカルリポジトリを理解しましょう
・パッチを送るときには、rebaseして、最新のパッチを送るようにしましょう
・mergeとrebaseは使い分けましょう。

・知っておくと良いこと
  -Gitオブジェクトの仕組み
  -indexの動き
  -コンフリクト時の復旧方法
  -git blame←誰がいじったか

■ Q&A

・過去の履歴を消すことはできるのか
  →消せるが、バージョン管理そのものが破綻する。
    分散型なので、みんなのパソコンに残る。
・index（昔は、cacheと呼んでた）は、簡単に言うと、次のコミット候補といえる。

・リビジョン番号はあるのか？
  →ハッシュ値しかない。
    ある時点での、名前（エイリアス）はtagで管理する。
  →とはいいながらも、40桁でやりとりするのは面倒なので、
    LKMLでも、最初の8桁ぐらいで話をすることが多い。

・ハッシュ値がぶつかることがあると思うが。
  →日付とか、名前の情報を入れて、それを圧縮したファイルのSHA1ハッシュ値を
    使うなどしている。

・Gitハブのような、Webフロントエンドはないか？
  CGitなど、何種類かある。

------------------------------------------------------------
Debianに新しいアーキテクチャを追加するには

岩松さん / ルネサス
------------------------------------------------------------
■ 岩松さん
・Debianに新しいアーキテクチャとして、SHを追加するために活動中
・本日は、Debianのビルドシステムとdebianパッケージの作り方

■ DebianProject
・自由なOSを目指すプロジェクト
・11個のアーキテクチャをサポート
・LinuxとFreeBSDをサポート
・ソフトウェアは同じバージョンでリリース
  AutobuilderNetworkが、全アーキテクチャのイメージをリリースしまくってる

■ Debianに追加するメリット
・ライセンスの問題がクリア
・パッケージポリシーが明確
・バグの集約、パッチ、アップストリームとの連携が密
・バイナリの提供、パッケージ配布のインフラが整備されている
・開発者の確保、CPUアーキテクチャサポート体制が整備されている。
  世界的にみれば、5000人ぐらいの開発者がいる。

■ Debianに追加するデメリット
・ソフトウェアバージョンの依存
  →新しいパッケージを導入するのは、メンテナ次第。

■ Debianのアーキテクチャサポート
・パッケージメンテナ
  -Debianパッケージのメンテナ
・Builddメンテナ
  -Autobuilder Networkとアーキテクチャ用のbuilddのメンテナンスチーム
    ★今回の話題
・リリースチーム
  -Stableリリースを管理するチーム
・セキュリティチーム
  →バグ、セキュリティ問題に対応する。

■ Autobuilder Network
・Package poolに、最新のソースと、パッチがためられている。
```

```
  Wanna-build＆databaseをbuilddメンテナが管理

・quinn-diffが、パッケージプールの状態を監視。
  差分があれば、「need build」フラグを立てる

    ここまでは、全アーキテクチャ共通システム。
    以下は、アーキテクチャ依存システム。

・Wanna-buildは、パッケージのビルド状態を管理
  エラー、ビルド中、依存パッケージ待ちなどを確認。

・これを、実際にビルドするデーモンが走っているマシンが、
  常にビルドをチェックしている。
  ビルド結果は、wanna-buildに通知する。

    →パッケージができたら、パッケージプールに置く。

・新しいアーキテクチャを追加するには、builddを備えたノードを設置する必要がある。

■buildd nodeを構築
・最初はクロスコンパイル
・debuild  dpkg-cross，devscripts，equivsを使って、クロス環境でパッケージを構築
・できないものはバグ報告
・Buildd nodeに必要な環境
  -build-essential
    ビルドに最低限必要なパッケージ（dpkg-dev，g++，libc-dev，make）
  -essental-packages
    ユーザー空間で必要なパッケージ郡
  -その他Xlibなどに依存する。

■Buildd node完了後
・Debian-ports MLで宣言
  新しいアーキテクチャを追加したいことを通知
  →DebianDeveloperのバックアップやこれまでの活動が大きく影響することもあり。
・debportsネットワークに追加される。
・あとは、自動でビルドされていく

■その他
・90%以上動かないと、stableリリースされない。
  現在、26000パッケージが提供されている。
・gccのテストは1週間ぐらいかかる。

■  まとめ
・昔はポーティングが大変でしたが、debportsによって新しい
   アーキテクチャサポートが大変でしたが、サポートが容易になりました。



------------------------------------------------------------
Latest Kernel Development Status Watch

小崎さん
------------------------------------------------------------
■  小崎さん
・カーネルWatchを書いてます。
・日本で数少ないLinuxカーネルジャーナリスト
・LKMLにもコミットしてます。
  -Signed-off-by数でTOP10%に入るぐらい
  -mm/ディレクトリ以下に限定すると10位
  -Reviewd-byでは7位ぐらい

■  カーネルの開発状況
・過去2年のコミット数推移
  2年で2倍。1万2千のパッチが1回のリリースで入る。
  →パッチ集計をcgiで更新して発表しているサイトがある。

・2.6.30のコミットレート
  -12000ぐらいのパッチ
  -組織別１８７の組織がコントリビュート。
    一位はHobbyists
  -日本企業の順位が上がってきた。
    Fujitsu，
```

```
   Renesas（SH関連），
   NTT（TOMOYO，ファイルシステム）
  -ディストリビューションベンダー、サーバーベンダーが強かったが、
   組み込み関連の伸びが目立つ。

■ マージウウィンドウ
・3ヶ月に1回コミット
・中心となるツリー(Linus)からツリーが各々分かれており、
  ネットワーク関連(davem/net-next-2.6)のものなど複数存在
・最初の2週間だけ、機能追加のパッチを受け入れる。
  それ以降は、バグ修正しか認めない。
・調べてみると、マージウィンドウで9割程度が投稿される。
 →今の開発サイクルは、うまくいっているといえる。
   まだまだ開発が加速している状況

■ サブツリー勢力図
・Linusツリーに献上するサブツリーたち

  davem/net-next（デービッド・ミラー1440）
   ネットワーク関連。無線関連が盛り上がっている。

  tipツリー（Ingoのツリー。1270）
   x86関連、ftraceまわり、redhatの都合などをマージしている。
   かなり広範囲。

  mmツリーなど（Andrew，975）

  mchehab
   V4L2

  tiwai
   Audio関連

  rmk

     →上位三大勢力と、各サブ機能ツリーという感じ

■ 最近の注目パッチ
・Compcache
  RAMの一部をSWAPとしてみせかける
  SWAPに書くときに、圧縮してから書く。
  SWAP-lessのシステムで、メモリ空間が広げられる。
  使ってないプロセスを退避させられるため、メモリ効率も良いはず。

  独自でメモリ管理をつくってしまったので、まだマージが見送られている。
  メモリ管理を見てるひとから、説明を求められている。

  目指しているところは合意がとれている。

  -開発者のホームページに、性能測定結果が掲載されている。
   Compcacheを使うと、プロセス増加に伴うメモリ増加がゆるやかになり、
   高負荷のときでもメモリがあいている。
   従来なら、メモリ不足で死んでる状況でも安定して動いている。

  -LZO圧縮
   圧縮時間の短さを優先

・カーネル圧縮の改善
  -Support LZMA/BZIP2
  -LZMAは、GZIPより30%圧縮率が向上。展開時間も、GZIPより速い。
   圧縮時に時間がかかるので、ユーザーのデメリットが無い。

・/dev/mem_notify on memcg
 →組み込み界隈からの投稿
 →気づかれずにスルーされている。。

   GregKHとgoogleのエンジニアは気にしているが、後回しにされている。

  -具体的な使用事例がちゃんと説明されたら入ると思う。
   組み込みで使われているが、表に出せないといわれて出てこない。

  -Linuxは互換性が非常に重視されている文化。
```

```
    クズ機能でも、必死でサポートする。
    どんどん、機能をオープンにして、議論に参加しましょう。

  -組み込み屋さんは、1行でも良いので、「good patch!」を返すこと。
    それが、機能を生かす方法である。
    この声明には意味がある。

・Per-bdi writeback flusher thread
  -bdi（backing device information）
    メモリ管理に必要な情報がピックアップされた構造体。
    デバイスごとに内容が異なる。RAMDISKなら、キャッシュの必要がないとか、
    tmpfsのダーティフラグ扱いなど、メモリ管理で参照する。
    ディスク1個につき、スレッドが1個できる。
  -いままでは、ディスク2-3個でも問題にならなかったが、
    最近は、SSDとディスク混合などのシステムで、性能が大幅に低下。
    全スレッドが一番遅いスレッドにつまる。
    デバイスごとのスレッド分割が基本的なアイデアだが、根本的な部分なので、
    リグレッションの発生が入念にチェックされている。
  -Andrewが性能系を心配しているが、懸念が解消されたら入るだろう。

・VFS based Union mount
  -stackableなファイルシステム
    ReadOnryとReadWriteの領域をうまく利用できるので、
    組み込みにメリットは多い。
  -AUFSはフレームウォーの結果、nackされた。入らない。
    UnionFSは、VFSをいじると、100種類以上ある
    Linuxのファイルシステムを変えるのに抵抗があった。
      →もうしばらく議論が必要

・reflink
  -ocfs2のシステムコールを追加したい。
    ファイルシステムから独立したcopy-on-writeの枠組み。
  -リンクコマンドでリンクをはるぐらいの速度でコピーできる。
    コピーパフォーマンスが向上する。

・devtmpfs
  -Devfs再び。
    過去のDevfsは、ユーザー空間との連携がうまくいかない。
  -udevは、起動時にデバイスをスキャンして、デバイスファイルを作っていく。
    やっぱり、RAMDISKで速くやりたい。
  -カーネルの初期化は、先にドライバが動かないとできない。
    デバイスファイル用のファイルシステムを提供するアイデアだが、
    またDevfsの再来かということで、フレームしている。


■　質疑
・小崎さんの最大のフレームウォーは？
  AppArmorが歴史的だと思う。
  LKMLは同じことを2回でてくるのを嫌う。
  TOMOYOは、過去にイメージの悪いパッチがあったので、たたかれた。

・LKML-embeddedにも投げたほうがよいか。
  自分では判断が難しい。いろいろ投げたほうが良い。
  Andrewも、どんどんコメントくれといっている。

・クリストフとかアルビロの性格は？
  彼らの発言の過去ログを見るとかが必要。
  Andrewはすごい良い人。
  反応してくれる人を指定するのが重要。

・mem_notifyのユースケースは、商品依存なので、非常に公開しにくい。
  デバイス依存の似たような機能は勝手に実装されている。
  一度、そういった話を対面でして合意を得られれば、一気に入るかも。

------------------------------------------------------------
最後に
------------------------------------------------------------

■次回のネタ
  フラッシュファイルシステムのお話とか

■カーネルサミット In 東京
```

```
・2009年10月
 年に一回のカーネル技術者の集まり。
 初のアジア開催

・誤解しないで！
 Linuxのロードマップが決まるわけではない。
 組み込みも当事者である。
  →組み込みの要望は、ちょっとずつで良いので、どんどん出してほしい。

・なぜ東京か
 -多様性を求めて。組み込み技術は注目されている。
 -日本の組み込みOSS技術者にとってチャンス

■ コミュニティーの意義
・現代の組み込み機器は高級なOSがもはや不可避
 -独自に作ってメンテナンスし、世界水準についていけるのか？

・開発者視点から
 -世界最高峰の技術と触れ合うチャンス
 -最高峰に見てもらうきっかけを作るチャンス
 -セッションの提案を積極的にチャレンジしませんか

■ その他
・LKML-embeddedとCELF-devの使い分けは？
 技術的なネタは、LKML-embeddedにどんどん投げてください。
 CELFdevは、そこに投げにくいものを気楽に投げてください。

・イベント開催のやり方などの提案もどうぞ。
 オンラインでつないで会場間をつなぐなど。
```

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWikiに残してください。

English Translation Volunteer

- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

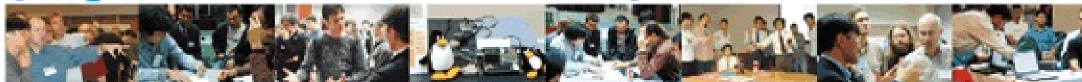
Categories:

- Japan Technical Jamboree
- Events

From: eLinux.org

Japan Technical Jamboree 29

CE Linux Forum
Japan Technical Jamboree



Date: July 17th / 日付: 7月17日（金）
At Nakano Sunplaza / 於、中野サンプラザ

Contents

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Linux Forum. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
  - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly without any complicated expressions. Most Japanese developers are capable to understand plain English.

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。
  - Japan Technical Jamboree 27 (Tokyo)
  - Japan Technical Jamboree 28 (Osaka)

Date and venue... / 日付・場所...

	• Date **July 17, 2009** ◦ **Starting at 10 am** • At ***Nakano Sunplaza*** / 会場 中野サンプラザ ◦ Nakano Sunplaza is located just close to **Nakano** station (JR/Tokyo Metoro). ◦ http://www.sunplaza.jp/ (Japanese) • Admission: ***Free of charge*** / 参加費用: 無料 • Not limited for CELF members. / CELF会員以外も参加・セッション持ち込み共に可能 • Coordinators / 世話役 (Your inquiries in English welcome) ◦ Hisao Munakata / 宗像尚郎 (munakata_dot_hisao(a)renesas_dot_com) ◦ Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

Registration / 参加登録

- Registration Page / 参加登録
  - Registration is only to forecast the approximate number of the attendees. No confirmation mail will be delivered. It is not mandatory.
  - 参加登録は大まかな参加人数把握のためにお願いしています。入力をされても確認メールは発信されません。参加登録をしないで出席していただいても構いません。

Main Topics

- Coming back from **Embedded Linux Conference 2009** and **Collaboration Summit**.

ELC 2009とCollaboration Summitの帰朝報告

Agenda / 進行

Agenda

Time	Title and presenter	Notes Presentation Materials
10:00am..	• Opening and Self Introduction / 連絡事項・自己紹介	
10:30am..	**Status of Embedded Kernel Features** (Tim Bird)	• Join jamboree from Montreal through Internet • Media:Status-of-embedded-kernel-features-2009-07.pdf • Video • English
11:15am..	**Live report from Montreal Linux Symposium** (Tim Bird)	• Join jamboree from Montreal through Internet • Media:LS-Canada-09-live-report.pdf • Video • English
11:30am..	**CELF Sponsoring Projects Status Update** (Tim Bird)	• Join jamboree from Montreal through Internet • Media:CELF-Contract-work-July09.pdf • Video • English
11:45am..	Lunch break	
1:00pm..	***Evaluation of Flash File Systems for Large NAND Flash Memory*** (Namihira, Toshiba)	• File:CELFJamboree29-FlashFS-Toshiba.pdf • Video
2:00pm..	***Latest Kernel Development Status Watch*** together with discussion about C Group, FTrace and so on(Kosaki)	**Video Session** • Video(Presentation) • Video(Discussion)
3:00pm..	**Introducing interesting topics from Japan Linux Symposium** (S. Ueda)	
3:45pm..	**Lightening Sessions**	• Video

- Please be noted above time table is just a guideline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶

対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWikiに残してください。

English Translation Volunteer

- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

Categories:

- Japan Technical Jamboree
- Events

> From: eLinux.org

Japan Technical Jamboree 3

Contents

Introduction

The Japan Technical Jamboree is a forum-wide technical meeting of the CE Linux Forum. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.

Coordinators: (Your inquiries in English welcome).

- Hisao Munakata (munakata.hisao(a)renesas.com)
- Satoru Ueda (Satoru.Ueda(a)jp.sony.com)

このサイトは原則として日本語のみです。 ***THIS PAGE was ;-) JAPANESE LANGUAGE ONLY***

Table Of Contents

History

第一回目の案内と、ミーティングメモなど(The First Jamboree: announcement, meeting memo)	Japan Technical Jamboree
第一回目の後に有志で行ったフォローアップミーティング(Follow up meeting after the First Jamboree)	Japan Technical Jamboree Follow Up
第二回目の案内と、ミーティングメモなど(The Second Jamboree: announcement, meeting memo)	Japan Technical Jamboree 2
韓国版ジャンボリー（第一回）(The First Korean Jamboree)	CELF Korea Tech Conference 純粋技術ネタだけでなく「コミュニティーへの加わり方」なども面白い(Not only pure technical staffs but "How to participate in communities" are interesting)
International Technilca Conference, June 2005	International Technical Jamboree

Latest Update

Date:	Item:
6 July	ディスカッションアイテムがすべて決まりました。(All discussion items fixed.)Agenda
20 June	参加登録サイト開設 (Registration site open)
20 June	開催告知メール発信(Announcemessage sent)

提案 / Proposal

- メールのやりとりについてですが、差し障り無ければ互いに「○○さん」としませんか？ :) (We would like to propose to call each other in mail rather frankly "XXX-san" than using square titles.)

重要連絡 / Important Notice

- 今後ジャンボリーの連絡は、Members@celinuxforum.orgメイリングリストではなく、celinux-dev@tree.celinuxforum.orgメイリングリストに発信します。このメイリングリストはCELFメンバー以外でも登録可能で技術情報のやりとりに使われます。第３回目以降のお知らせもそちらのメイリングリストでさせていただきますので是非ご登録ください。(From now on, the messages concerning Jamborees will be sent to celinux-dev@tree.celinuxforum.org, not to Members@celinuxforum.org. Non-members of CELF can subscribe to this mailing list for technical discussion. Announcements of following Jamborees will also sent to new mailing list and we recommend you to subscribe to it.)
- 登録はこちらで出来ます。(You can subscribe here.)

第３回　CELF テクノジャンボリー・東京 / The Third CELF Techno Jamboree Tokyo

- コーディネーター**(Coordinators)**：
  - 宗像尚郎 / Hisao Munakata
    - munakata.hisao@renesas.com
    - TEL: 03-3861-2642
  - 上田　理 / Satoru Ueda
    - satoru.ueda@jp.sony.com
    - TEL: 03-5435-3703

参加方法 / How to Participate

1. 参加登録をしてください。参加登録サイトはこちらに有ります。(Please register to attend. Registration site is here.)
    - なお、参加登録サイトはCELFメンバー以外はアクセスできません。メンバー以外の方は恐れ入りますがコーディネータに氏名、所属企業・大学等の情報、及び電子メールアドレスをお知らせください。登録を代行します。または周囲にフォーラムメンバーの方がおりましたら登録代行をお願いしてください。(Only CELF members can access to the registration site. For non-members, please let the coordinator know your name, company/university and e-mail address. The coordinator will register on behalf of you. Or, if you can find a member, ask him/her to register for you.)
    - CELFメンバー企業の社員（構成員）の方は、こちらで個人登録をしてください。なお、CELFの会則に拠ればメンバー企業が50%丁度、またはそれ以上の株式を所持している系列会社も会員と見なされます。(For employees of CELF member companies, please register individually here. According to the bylaws of CELF, subsidiaries 50% ore more of whose stock is owned by a member company is also regarded a member.)
2. 「ネタ振りカード」を記入してコーディネータ宛にお送りください。なるべく日本時間で前日（7月14日）午前中までに届くようにお願いします。(Fill in Neta-furi (What I can talk about) card and send it to the coodinator, so that the coodinator receive by AM of the previous day (July 14), if possible.
3. 当日は受付等有りません。会場にそのままお入りください。(There is no reception. Come up to the meeting room directly.)
4. ディスカッションテーマ登録歓迎です。このWikiページのAgenda欄に直接記入してください。またはコーディネータにお問い合わせください。(We welcome the Discussion Topics. Please fill in directly in the Agenda field of this Wiki page. Or contact one of the coordinator.)
    - なお、その際にはあらかじめ興味の有りそうな方と連絡をとっておいて頂き活発な意見交換が出来るよう準備いただければ幸いです。(It would be appreciated if you contact beforehand those who might be interested in that topic so that they can prepare and have active interaction at the Jamboree.)

ネタ振りカード / "Neta-furi" (What I can talk about) Card

- **Download:** Media:selfIntro.ppt
    - 今回は参加者は「ネタ振りカード」に必要事項を記入の上、あらかじめコーディネータにお送りください。(For this event, the attendees should fill in "Neta-furi" (What I can talk about) card and send it to the coordinator in advance.)
        - Power Pointまたはpdfファイルでコーディネータまで送ってください。(The card should be in Power Point or PDF file.)
        - 「ネタ振りカード」が書けないから出席できない、では本末転倒です。どうしても書けない場合はその旨をメールでコーディネータにお伝えください。(If you think you can not attend because you can not write "Neta-furi" card, it is absurd. In that case, mail so to the coordinator.)
    - ジャンボリー終了直後にみなさんに配布いたします。（配布を希望されない方はカードのチェック欄に必ずチェックしてください）。(Collected cards will be distributed to the attendees after the Jamboree. If you do not want your card distributed, please check the checkbox on the card.)
    - CELFはエンジニアが相互に情報交換をし会う場です。「ネタ振りカード」はそのきっかけを作ろうとするものです。(CELF is a place where engineers exchange information each other. "Neta-furi" cards are intended to be a trigger for it.)
    - 現在取り組んでいる開発テーマや興味のあるテーマなど、フリーフォームでご記入ください。(Write down,in free form, the topics of development you are now working or the topics you have interest.)
    - 冒頭にネタフリカードをもとに簡単に自己紹介していただきます。なお、カードの参加者への配布を希望されない方は自己紹介を省く事も可能ですが、フォーラムの趣旨をご理解いただき差し障りの無い範囲で結構ですので一言で結構ですのでご紹介いただけるようお願いします。(At the beginning, the attendees would briefly introduce themselves following to the "Neta-furi" cards. In the case you do not want your card to be distributed, though you may decline your self introduction, we would like you to understand the intention of the CELF and to introduce yourself to the extent possible.)
- 書き方(How to write)

- 名前(Name)メールアドレス(Mail Address)は必ずお書きください。万一メールアドレスが無いまたは一切開示出来ない場合は空欄でもやむを得ません。名刺を添付できる場合はそれに代えて結構です。(Anybody should fill in Name and Mail Address fields. In case that you have no mail address or do not want it public, the field may be left blank. You can attach your name card instead.)
- "Your Project or Interest"の部分には現在開発活動に協力中のプロジェクトまたは興味のある開発プロジェクトを記入してください。この部分を見て、同様な興味を持たれている方からコンタクトが有るかも知れません。(In the "Your Project or Interest" field, write in the projects that you are participating in development activities or the projects you are interested in. Seeing this field, people with the similar interest might contact you.)
  - 無論のこと社外秘等を冒してまで記入していただく必要は有りません。(Of course, do not take risks to disclose anything confidential.)
  - 更に広くパートナーを求めたい方はCELFのProjectList等でも紹介が可能です。右下隅にチェック欄が有りますのでチェックしてください。なおProject List はCELFメンバー専用のWiki ページにも有ります。(If you want further to seek partners, we can list your project on the ProjectList of CELF. Check the box on bottom right of the card. There is another project list in the wiki page for CELF members only.) ProjectList
- 写真は無くてももちろん結構です。(Photo need not be attached, of course.) ;)

- 使い方(How to Use)
  - 配布された「ネタ振りカード」の中で興味のある項目を見つけましたら是非コンタクトしてみてください。このカードはそのような関係作りのきっかけになることを目的にしています。(Try to contact the person if you find any interesting items in the distributed cards. These cards are intended to trigger such new relationship.)
- 従来お願いしておりましたショートプレゼンテーションは今回は行いません。より気軽に参加いただけるようになったかと思いますが如何でしょうか。(Unlike previous events, there will be no short presentation. You feel easier, don't you?)

Registration

- 参加登録サイトを開設しました。登録サイト (Registration Site is open.
  - なお、登録いただきましたデータはCE Linux Forumのアメリカに設置してありますサーバに保管されます。今回のイベントの出席確認、並びに参加登録者のみに宛てた今回のイベントに関する案内送信以外には使いません。また、イベント終了後資料送付が終わり次第データを消去致します。(Data entered as you register is saved in the server of the CE Linux Forum located in the US. This data will be used only for checking the attendees and sending messages to guide this event to registered attendees. After the event, the data will be erased as soon as meeting materials are sent to the attendees.)

Date and Location

- 日付。(Date)
  - 7月15日（金）10:00 AM ～ 5:00 PM (July 15 (Fri) 10:00-17:00)
- 場所(Location)
  - 中野サンプラザ・研修室２ (Nakano Sun Plaza, Lecture room 2)
    - http://www.sunplaza.jp/
    - JR中央線、中野駅前 (Just in front of Nakano station, JR Chuo line)
- お願い(Please)
  - 会場では商用電源のコンセントが充分に確保できない可能性が有ります。恐れ入りますがテーブルタップをお持ちの方はご持参いただければ助かります。(There may not be enough power outlets in the meeting room. It would be grateful if you attend with your extension cable.)
- 服装はカジュアルでどうぞ。(Dress code is Casual".)
- 会場定員が90名です。90名を越えた方は床に座っていただく等あり得ますのでジーンズ等がよろしいかと。 ;) (Capacity of the meeting room is 90 people. When exceeding the capacity, you may have to sit on the floor. We recommend jeans etc. ;))
  - エンジニア以外（メディア等）からご出席頂く方にお願いが有ります。今回は会場が狭い関係で、立ち席をお願いする可能性が有りますこと、恐縮ですがあらかじめご承知おきください。(To non-engineer attendees (Press etc.): we may ask you to attend standing due to the small room, sorry in advance).

ショートプレゼンテーション / Short Presentation

- 参加される方は、ショートプレゼンテーションをお願いします。これは「ネタ振りカード」をスクリーンに映してそれをもとに自己紹介していただく形にします。一言で結構です。(For attendees, please give a short presentation. In this event, each attendee introduces oneself with the "Neta-furi" card projected on the screen. It can really be short.)
- 前回のジャンボリーから若干時間が経っていますので、今回改めて全員にお願いします。(As time has elapsed since the last Jamboree, we ask all the attendees to give short presentation.)
    - どのような技術分野に関してどのようなパッチをご提供いただけるか、どのような技術分野のいかなる事で開発に難儀しているか、どのような事に関してお互いに情報交換したいか・・・をお互いに知る手がかりにしたいと思います。(We hope this to be a clue for all to know what patches you can submit in what technical field, what issue you are struggling with, or on what topics you would like to exchange information each other.)
    - ご承知の通り、CELFはGPL/LGPLないしはそれらに互換性があるライセンスによって自由に交換できるソフトウエアをベースに技術ディスカッションを進める場です。上記に関しましてもその条件を前提としてお話しいただければと思います。(As you know, CELF is a place for technical discussion based on software freely exchangeable with GPL/LGPL or compatible licenses. Please talk with that assumption.)
- プレゼンテーションの内容に関して一切のお約束等は無くて結構です。No commitment, No obligationで結構です。(You do not have to commit anything on the content of your presentation. No commitment/No obligation.)

Agenda

AGENDA

Time

Item

10:00..10:30

自己紹介セッション (self introduction session)

10:30..12:00

携帯電話に**Linux**を実装する！ 水山さん（パナソニックモバイル）(Implementing Linux on mobile phone Mizuyama(Panasonic Mobile))

12:00..13:00

(昼食：各自でお願いします) (Lunch: Find your own lunch :-))

13:00..14:00

ARTLinux, ハードリアルタイム処理機能を拡張した**Linux**カーネル (ARTLinux, Linux kernel enhanced with hard realtime feature

14:00..15:00

Fast Boot/Shutdown with Power Management Function (Snapshot Boot)

15:00..16:00

Linux Tinyの分析 (Analysis of Linux Tiny)

16:00..16:30

自己紹介セッション (遅れて参加した方向け) (self introduction session for late comers)

16:30..17:00

フォーラムの現況報告（この前までのセッションが時間通り終わらなかった場合は省略します）(Update from CELF (may be cancelled, if not on schedule))

デモンストレーション / Demonstrations

- 未定 (TBD)

メインセッション / Main Session

Making Mobile Phone with CE Linux

- 6月13日に開催されたインターナショナルテクニカルカンファレンスでパナソニックモバイルの水山さんのセッションがおおいに注目を集めました。今回はこのセッションを日本語でもう一度繰り返し、ディスカッションを深めたく思います。(Mr. Mizuyama's session was very successful at the International Technical Conference held on June 13. The session will be rerun in Japanese, and we hope to have more discussion.)
  - International Techinical Conference・・・International Technical Jamboree
  - 水山さんのプレゼン資料はここにあります(Mizuyama's presentation material)・・・ITJ2005Detail1-2

ARTLinux, ハードリアルタイム処理機能を拡張したLinuxカーネル / ARTLinux, Linux kernel enhanced with hard real time processing feature

- ムービングアイ、石綿さん（メールを下記コピー&ペーストします）(by Ishiwata, MovingEye Inc.. His message copied and pasted below:)

```
内容(content):
Linuxカーネルにリアルタイム処理機能を拡張する試みはいくつかありますが,
それらと比較しながらARTLinuxにおける拡張手法とその実装について説明し,
性能測定の結果を示します.(There have been several attempts to enhance
Linux kernel with realtime processing feature. Comparing with them,
we will explain our enhancement in ARTLinux and its implementation
and show the results of performance measurement.)

リアルタイム処理を割込処理に基づいて実装する事例が多いと考えますが,
ARTLinuxでは周期処理に基づいて実装する手法を推奨しています.ですので,周
期処理に基づく実装の是非について議論できればと思います.(While in many
cases the realtime features are implemented based on interrupt processing,
in ARTLinux we recommend to implement based on periodic processing.
Therefore we hope to discuss pros and cons of implementation based on
periodic processing.)
```

- プレゼンテーション資料(presentation material) Media:ARTLinuxIntro.ppt

Fast Boot/Shutdown with Power Management Function

- ソニー・町田さん(by Machida, Sony)
- この改善は水山さんのプレゼンテーションにも深く係わることが有ります。(This improvement may strongly be related to Mizuyama's presentation.)
- 内容に関してはこの資料を参照してください。(See the material for the contents.)
  - Media:20050614-snapshot-boot.-pm.pdf

Linux Tinyの分析 / Analysis of Linux Tiny

- NEC・池田さん(by Ikeda, NEC)
- Linux Tiny は、カーネルのサイズ縮小・使用メモリ削減のためのパッチ集です。このセッションでは、NEC Linux推進センターがでこれまでに行った Linux Tiny の内容分析についてご紹介します。(Linux Tiny is a collection of patches to reduce kernel size and to save memory usage. In this session we introduce the analysis of Linux Tiny done so far at our Linux Promotion Center, NEC.)
  - Media:LinuxTiny_celfJtechJ3.pdf
  - Media:linutiny_celfJtechJ3_en.pdf
    - English translation by S. Ueda (BAD QUALITY)

そのほか / Others

FAQ

このジャンボリーに何か特別なルールは有りますか？ / Is there any special rule for this Jamboree?

- **3N** ルールが有ります。3Nとは・・・、(We have "3N" rules, where 3N means:)
    - No Confidentiality: 守秘義務は一切有りません。
    - No Obligation: 義務や責任は一切有りません。
    - No Commitment: 約束無しで構いません。
- CELFはオープンコミュニティーです。"**Open Source Community Way**"で！ :) (CELF is an open community. We do "Open Source Community Way" :))

参加者はCELFメンバーに限られますか？ / Is an attendee restricted to CELF member company?

- いえ、メンバー以外の方の参加も歓迎します。(No, non-members are also welcome.)
- なお、このWikiページはCELFメンバー以外の方もアクセス可能です。但し、参加登録サイトはCELFメンバーのみアクセス可能です。CELFメンバー以外の方が参加を希望される場合は、参加方法の項目を参照してください。(Non-members of CELF can also access this Wiki page. But only CELF mebmers can access to the registration page. For non-members to attend, see the "How to Participate" above.)

ショートプレゼンテーションは必須ですか？ / Is the short presentation mandatory?

- 是非！今回は試しに「ネタ振りカード」で効率、効果アップを試します。(Yes, please In this event we attempt to improve efficiency and effectiveness by using "Neta-furi" cards.)
    - もちろん、No obligation / No commitmentで結構です。(Of course, no problem with "No obligation / No commitment".)

日本語がわからない方も参加できますか？ / Can non-Japanese speaker attend?

- 参加いただいても結構です。但し、翻訳サービス等はございません。(Yes, you can attend. But we can not provide translation service etc..)

会費はいくらですか？昼食等は用意がありますか？ディナーパーティーは？ / How much is the fee? Lunch prepared? Dinner Party?

- 会費は無料です。(The fee is free.)
- 昼食・ディナーパーティー等の用意は有りません。(No lunch nor dinner party prepared.

デモンストレーションをしても構いませんか？ / Can I demo?

- 商品（ソフトウエアツール、開発ツールなど）のデモンストレーションは昼休みの時間にお願いします。(Please demo products (software tools, development tools) at lunch break.)
- 開発成果物のデモンストレーションに関しては、ご要望になるべくこたえられるようにしたく思います。早めにコーディネーターにご連絡ください。(As to demonstration of your development work, we would like to meet your request as much as possible. Please contact the coordinators as early as possible.)

他の地域でも開催されますか？ / Is the event held in other regions?

- 2005年5月に韓国でも開催。300人近い参加者を集め極めて盛況だったようです。(The Jamboree was held in Korea in May 2005. It is reported to be very successful with about 300 attendees.)

イベント開催を待たずに技術検討を始めたい / I can't wait the event and want to start technical study now

- 特定の技術テーマに絞ったWikiページは、今すぐにでも設定可能です！また、メイリングリストの設置も可能です。(We can immediately set up mailing list or Wiki page for specific technical field.)
- celinux-dev@tree.celinuxforum.orgメイリングリストの活用をお勧めします。(We recommend to use the mailinglist celinux-dev@tree.celinuxforum.org.)
    - こちらはメンバー以外の方も登録可能です。CE機器向けLinux に関係する方ならどなたでも参加できます。(This mailing list is open to non-members. Anybody involved in Linux for CE equipments can subscribe.)
    - 日本語のメールも配信可能です。日本語圏外の登録者も大勢居ますので、日本語メールには英文のサマリーを付けてください。ごく簡単で構いません。(You can send messages in Japanese. But please add a summary, if short, in English as many of non-Japanese speakers are also subscribing.)

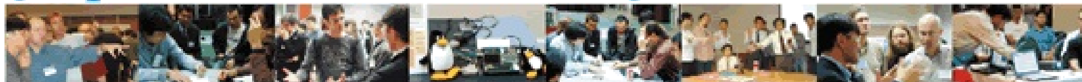
Categories:

- Events
- Japan Technical Jamboree

From: eLinux.org

Japan Technical Jamboree 30

CE Linux Forum
Japan Technical Jamboree



Date: October 2nd / 日付: 10月2日（金）
At Nakano Sunplaza / 於、中野サンプラザ

Contents

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Linux Forum. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
  - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly without any complicated expressions. Most Japanese developers are capable to understand plain English.

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。
  - Japan Technical Jamboree 29

Date and venue... / 日付・場所...

<table>
<tr>
<td></td>
<td>

- Date **October 2nd, 2009**
  - **Starting at 10 am**
- At ***Nakano Sunplaza*** / 会場 中野サンプラザ
  - Nakano Sunplaza is located just close to **Nakano** station (JR/Tokyo Metoro).
  - http://www.sunplaza.jp/ (Japanese)
- Admission: ***Free of charge*** / 参加費用: 無料
- Not limited for CELF members. / CELF会員以外も参加・セッション持ち込み共に可能
- Coordinators / 世話役 (Your inquiries in English welcome)
  - Hisao Munakata / 宗像尚郎 (munakata_dot_hisao(a)renesas_dot_com)
  - Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

</td>
</tr>
</table>

Registration / 参加登録

- Registration Page / 参加登録
  - Registration is only to forecast the approximate number of the attendees. No confirmation mail will be delivered. It is not mandatory.
  - 参加登録は大まかな参加人数把握のためにお願いしています。入力をされても確認メールは発信されません。参加登録をしないで出席していただいても構いません。

Main Topics

- Just before the Japan Linux Symposium!

Japan Linux Symposium直前です！

Agenda / 進行

Agenda

- Video archive will be prepared aroundOctober 31.

Time

Title and presenter

Notes

Presentation Materials

10:00am..

- Opening / 連絡事項

10:15am..

Connected from USA

- **Tim Bird**
  - Latest Community topics update
    - Video

  - Latest CELF development project update
    - Video
  - Brief Introduction of Japan Linux Symposium
    - "Measuring Function Duration with Ftrace"
      - Video
    - "Embedded Developer Bof"
- **Frank Rowand**
  - Brief Introduction of Japan Linux Symposium
    - "Real-Time Linux Failure"
      - Video
- Join jamboree from USA through Internet
- English
- Media:CELF-status-oct-2009.pdf
- Media:State-of-embeddedLinux-oct-2009-update.pdf
- Media:Measuring-function-duration-with-ftrace-oct01.pdf

11:45am..

Lunch break

1:00pm..

Voice from embedded system developers to Kernel Summit participants (Fernando Luis Vázquez Cao, NTT Data)

- Kernel Summitの中に組込みシステムユーザーの声を聞く場がありそうです。そこでどのような話を聞いてもらいましょうか？オープンディスカッションです。
- This session is an open discussion. In the coming Kernel Summit, there would be arranged a session listening to the Linux users in embedded system field. We'd like to discuss what we should tell in the opportunity.
- Video

2:00pm..

Evaluation of UBI and UBIFS (Katsuki Uwatoko)

- Media:CELFJamboree30-UBIFS_update.pdf
- Video

3:00pm..

Preview of Japan Linux Symposium (TBD)

4:00pm..

Calling for your session proposals

- Session proposal how-to. / 提案の方法
- Please be noted above time table is just a guideline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWikiに残してください。

English Translation Volunteer

- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

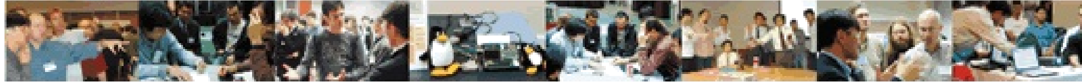
Categories:

- Japan Technical Jamboree
- Events

> From: eLinux.org

Japan Technical Jamboree 31

CE Linux Forum
Japan Technical Jamboree



Date: December 18th / 日付: 12月18日（金）
At Nakano Sunplaza / 於、中野サンプラザ

Contents

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Linux Forum. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
  - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly without any complicated expressions. Most Japanese developers are capable to understand plain English.

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。
  - Japan Technical Jamboree 30

Date and venue... / 日付・場所...

<table>
<tr>
<td></td>
<td>

- Date **December 18th, 2009**
  - **Starting at 10 am**
- At ***Nakano Sunplaza*** / 会場 中野サンプラザ
  - Nakano Sunplaza is located just close to **Nakano** station (JR/Tokyo Metoro).
  - http://www.sunplaza.jp/ (Japanese)
- Admission: ***Free of charge*** / 参加費用: 無料
- Not limited for CELF members. / CELF会員以外も参加・セッション持ち込み共に可能
- Coordinators / 世話役 (Your inquiries in English welcome)
  - Hisao Munakata / 宗像尚郎 (munakata_dot_hisao(a)renesas_dot_com)
  - Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

</td>
</tr>
</table>

Registration / 参加登録

- Registration Page / 参加登録
  - Registration is only to forecast the approximate number of the attendees. No confirmation mail will be delivered. It is not mandatory.
  - 参加登録は大まかな参加人数把握のためにお願いしています。入力をされても確認メールは発信されません。参加登録をしないで出席していただいても構いません。

Main Topics

- Final jamboree of 2009!

今年最後のジャンボリーです！

Agenda / 進行

Agenda

Time	Title and presenter	Notes Presentation Materials
10:00am..	• Opening / 連絡事項	• Media:SelfIntro31.pdf
10:15am..	**Connected from USA** • **Tim Bird** ◦ Latest Community topics update	• Join jamboree from USA through Ir • Media:State-of-embedded-Linux-de update.pdf • Video ◦ (English / 英語のセッションで
	• **Tim Bird** ◦ Latest CELF development project update ▪ Incruding brief introduction of current Open Project Proposals	• Media:CELF-project-info.pdf • Video ◦ (English / 英語のセッションで
12:30pm..	• **Inagaki-san (Kuroto-shiko / Buffalo)** ◦ Brief demonstration of SheevaPulg (Kuro-Sheeva)	• Video

1:00pm..	• **Fujii-san (Notava Japan)** ◦ Introduction of NoTA project	• Media:Notava_NoTA_Intro_200912
1:45pm..	• **Munakata (Renesas)** ◦ [JLS replay] Rescuing SuperH to Linux common place	• Video
2:30pm..	• **Kawasaki-san (Hitachi)** • ***Android Related (1)*** ◦ Porting Android to SuperH platform	• Presentation Material (pdf file) • Video • Related demonstration Video
3:15pm..	• **Matsubara-san (IGEL)** • ***Android Related (2)*** ◦ [ET2009 enhanced] Utilize SoC built-in video accel. engie on Android	• Presentation Material (pdf file) • Video • Related demonstration Video
4:00pm..	• **Kobayashi-san (KMC)** • ***Android Related (3)*** ◦ Let's play with goldfish ▪ How to build android emulator. And quick review of Eclair source release. ▪ http://blog.kmckk.com/archives/1998269.html	• Video
4:45pm..	• **Kobayashi-san (Toshiba)** ◦ Evaluation of Data Reliability on Linux File Systems ▪ 同期モードでファイルに書き込めばデータは守られるのか？という観点から，実験データをもとに発表します・ ▪ Is the synchronization mode for writing file really protect the data? Discussion based upon experiments.	• Presentation Material (pdf file) • Video
5:30pm..	• **Handa-san (NTTデータ先端技術)他** ◦ Status report of TOMOYO Linux ▪ TOMOYO Linuxプロジェクトの近況についてご紹介します	• Video
CANCELED Due to influenza	• **Kanda-san (SDY)** ◦ Lock free Algorithm for C++ Container ▪ Lock freeアルゴリズムを使用したC++言語のコンテナを紹介し、マルチコアCPUでのマルチスレッド性能などを発表します。 ▪ Introduction C++ Container used lock free algorithm,and discussion based multi-thread performance on multi-core CPU.	

- Please be noted above time table is just a guideline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWikiに残してください。

English Translation Volunteer

- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

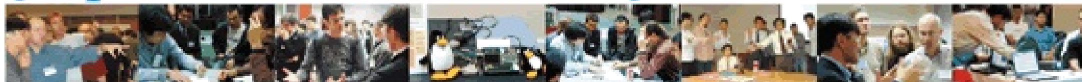
Categories:

- Japan Technical Jamboree
- Events

> From: eLinux.org

Japan Technical Jamboree 32

CE Linux Forum
Japan Technical Jamboree



Date: March 5th / 日付: 3月5日（金）
At Nakano Sunplaza / 於、中野サンプラザ

Contents

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Linux Forum. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
  - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly without any complicated expressions. Most Japanese developers are capable to understand plain English.

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。
  - Japan Technical Jamboree 31

Date and venue... / 日付・場所...

	• Date **March 5th, 2010** ◦ **Starting at 10 am** • At ***Nakano Sunplaza*** / 会場 中野サンプラザ ◦ Nakano Sunplaza is located just close to **Nakano** station (JR/Tokyo Metoro). ◦ http://www.sunplaza.jp/ (Japanese) • Admission: ***Free of charge*** / 参加費用: 無料 • Not limited for CELF members. / CELF会員以外も参加・セッション持ち込み共に可能 • Coordinators / 世話役 (Your inquiries in English welcome) ◦ Hisao Munakata / 宗像尚郎 (munakata_dot_hisao(a)renesas_dot_com) ◦ Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

Registration / 参加登録

- Registration Page / 参加登録
  - Registration is only to forecast the approximate number of the attendees. No confirmation mail will be delivered. It is not mandatory.
  - 参加登録は大まかな参加人数把握のためにお願いしています。入力をされても確認メールは発信されません。参加登録をしないで出席していただいても構いません。

Main Topics

- The first jamboree of 2010!

今年最初のジャンボリーです！

Agenda / 進行

Agenda

Time

Title and presenter

Notes

Presentation Materials

10:00am..

- Opening / 連絡事項

10:15am..

Connected from USA

- **Tim Bird**
  - Latest Community topics update
- Join jamboree from USA through Internet

  - (English / 英語のセッションです)
- Presentation Material (pdf file)
- Video
  - (English / 英語のセッションです)
- **Tim Bird**
  - Latest CELF development project update
    - Including brief introduction of current Open Project Proposals
- Join jamboree from USA through Internet
  - (English / 英語のセッションです)
- Presentation Material (pdf file)
- Video
  - (English / 英語のセッションです)

11:30am..

Lunch

12:30pm..

Introducing the bootloader 'barebox'

- **Wolfram Sang (Pengutronix)**
- Presentation Material (pdf file)
- Video
  - (English / 英語のセッションです)

1:30pm..

Using QEMU for cross development

- with demonstration
- **Tetsuyuki Kobayashi (KMC)**
- http://blog.kmckk.com/archives/2363482.html
- Presentation
  - Video
- Demonstration
  - Video

2:30pm..

Lock free Algorithm for C++ Container

- **Hiromasa Kanda (SDY)**
- Presentation Material (pdf file)
- Video

3:30pm..

Debian and embedded system

- **Nobuhiro Iwamatsu (Renesas)**
- Video

4:30pm..

Long-term testing on accelerated Linux kernel

- **Yoshitake Kobayashi (Toshiba)**
- 長期稼動テストを実施するに当たり，Linuxカーネルの内部時間を実際に加速してみたという話です
  - (少し怪しい部分はあります) :-)
- A challenge to increase the kernel internal time clock to simulate long-term run.
  - Some weird item included. :-)
- Presentation Material (pdf file)
- Video

5:00pm..

- ***Calling for your session proposals***
- Session proposal how-to. / 提案の方法
- Please be noted above time table is just a guideline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWikiに残してください。

English Translation Volunteer

- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

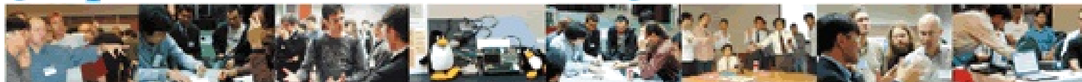
Categories:

- Japan Technical Jamboree
- Events

From: eLinux.org

Japan Technical Jamboree 33

CE Linux Forum
Japan Technical Jamboree



Date: June 4th / 日付: 6月4日（金）
At Nakano Sunplaza / 於、中野サンプラザ

Contents

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Linux Forum. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
  - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly without any complicated expressions. Most Japanese developers are capable to understand plain English.

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。
  - Japan Technical Jamboree 32

Date and venue... / 日付・場所...

<table>
<tr>
<td></td>
<td>

Date June 4th, 2010
Starting at 10 am
At Nakano Sunplaza / 会場 中野サンプラザ
Nakano Sunplaza is located just close to Nakano station (JR/Tokyo Metoro).
http://www.sunplaza.jp/ (Japanese)
Admission: Free of charge / 参加費用: 無料
Not limited for CELF members. / CELF会員以外も参加・セッション持ち込み共に可能
Coordinators / 世話役 (Your inquiries in English welcome)
Hisao Munakata / 宗像尚郎 (munakata_dot_hisao(a)renesas_dot_com)
Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

</td>
</tr>
</table>

Registration / 参加登録

- Registration Page / 参加登録
  - Registration is only to forecast the approximate number of the attendees. No confirmation mail will be delivered. It is not mandatory.
  - 参加登録は大まかな参加人数把握のためにお願いしています。入力をされても確認メールは発信されません。参加登録をしないで出席していただいても構いません。

Main Topics

- Bring the exitement of ELC to Japan!

ELCの興奮を日本に持ち帰りましょう！

Agenda / 進行

Agenda

Time	Title and presenter	Notes Presentation Materials
10:00am..	• Opening / 連絡事項	
10:15am..	**Connected from USA** • **Tim Bird** ◦ Latest Community topics update	• Join jamboree from USA through Internet ◦ (English / 英語のセッションです) • Media:State-of-embedded-Linux-June-2010-update.ppt • Video
	• **Tim Bird** ◦ Latest CELF development project update ▪ Including brief introduction of current Open Project Proposals	• Join jamboree from USA through Internet ◦ (English / 英語のセッションです) • Media:CELF-project-info-June-2010.ppt • Video

11:30am..	Lunch	
12:30pm..	**ELC-2010 Reoprt** • S. Kato (Panasonic)	• Media:ELC2010_Report_Jamboree.ppt • ELC2010 Presentations • Video
1:30pm..	**Kexec - Ready for Embedded Linux?** • Magnus (Renesas) ◦ (Repeat of ELC-2010 session)	• PDF • (English / 英語のセッションです) • Video (main) • Video (demonstration)
2:30pm..	**Google I/O 参加報告 / Reporting "Google I/O"** • Sim Chin Yeow (Renesas)	• Video
3:30pm..	**Verification of response time in various Real-time implementation** • Kouta Okamoto (Toshiba) ◦ RT拡張数種類を使って，周期タイマのレイテンシを測定しました ◦ A challenge to verify cyclic latency in various Real-time implementation	• Media:Verification_of_response_time-20100604.pdf • Video
4:15pm..	**Cross-building on user mode QEMU** • Tetsuyuki Kobayashi (KMC)	• http://d.hatena.ne.jp/embedded/20100508/p1 • Video
5:00pm..	**OBSとターゲットHWクラスターを活用したネイティブビルドシステムについて/ Native Build System using OBS and Target Hardware Cluster** • Hideto Kobayashi (Nomobok)	• Video
5:30pm..	**Evaluation of busybox bootchartd** • K. Yasui (Toshiba)	• Media:bootchartd-2010-604.pdf • Video
6:00pm..	**Understanding Froyo DalvikVM JIT (Japanese translation of Google I/O session)** • T. Kobayashi (KMC)	• http://www.slideshare.net/tetsu.koba/froyo-jit-20100605 • Video

- Please be noted above time table is just a guideline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWikiに残してください。

English Translation Volunteer

- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

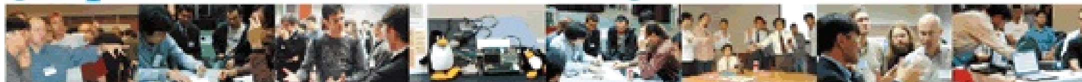
Categories:

- Japan Technical Jamboree
- Events

> From: eLinux.org

Japan Technical Jamboree 34

CE Linux Forum
Japan Technical Jamboree



Date: September 3rd / 日付: 9月3日（金）
At Nakano Sunplaza / 於、中野サンプラザ

Contents

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Linux Forum. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
  - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly without any complicated expressions. Most Japanese developers are capable to understand plain English.

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。
  - Japan Technical Jamboree 33

Date and venue... / 日付・場所...

	• Date **September 3rd, 2010** ◦ **Starting at 10 am** • At ***Nakano Sunplaza*** / 会場 中野サンプラザ ◦ Nakano Sunplaza is located just close to **Nakano** station (JR/Tokyo Metoro). ◦ http://www.sunplaza.jp/ (Japanese) • Admission: ***Free of charge*** / 参加費用: 無料 • Not limited for CELF members. / CELF会員以外も参加・セッション持ち込み共に可能 • Coordinators / 世話役 (Your inquiries in English welcome) ◦ Hisao Munakata / 宗像尚郎 (munakata_dot_hisao(a)renesas_dot_com) ◦ Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

Registration / 参加登録

- Registration Page / 参加登録
  - Registration is only to forecast the approximate number of the attendees. No confirmation mail will be delivered. It is not mandatory.
  - 参加登録は大まかな参加人数把握のためにお願いしています。入力をされても確認メールは発信されません。参加登録をしないで出席していただいても構いません。

Main Topics

- It is just before the LinuxCon Japan!

LinuxCon Japan直前です！

Agenda / 進行

Agenda

Time

Title and presenter

Notes

Presentation Materials

10:00..

- Opening / 連絡事項

10:15..

Connected from USA

- **Tim Bird**
  - **Improving Android Bootup Time**
  - Regular topics "Latest Community topics update" will be presented in the upcoming LinuxCon Japan.
- Join jamboree from USA through Internet

  - (English / 英語のセッションです)
- Media:Android-bootup-time-celf-jamboree34.ppt
- Video

11:30..

Lunch

12:30..

- **T. Kobayashi (KMC)**

  - **Logging system of Android**
- slides

- Video
- Video

13:15..

- **T. Kobayashi (KMC)**
  - **Reusing your existing software resource on Android**
    -
      1. Android上で既存のソフトを動かす Runnig the existing software on Android
      2. 1.1 Android用ツールチェインで再ビルド Rebuilding by Android tool chain
      3. 1.2 再ビルドせずに既存のバイナリをそのまま動かす Running the binary as is
    -
      1. 既存のLinux環境の中でAndroidを動かす Running Android on your existing Linux environment
      2. ARM Ubuntu 10.04(GUI無し)の上でAndroid 2.2を動かすことに成功したのでその方法の紹介 I show you 'Android on Ubuntu server'

slides

- related page for 1.1(in Japanese)
- related page for 1.2(in Japanese)
- related page for 2(in Japanese)
- Video
- Video

14:00..

- **Y. Iwamatsu (Renesas)**

  - **Debian Conference 2010 参加報告**
- Video

14:30..

- **Y. Kobayashi and K. Okamoto (Toshiba)**

  - **Evaluation of response time with memory access load**
- Video

- Media:EvaluationOfResponseTimeWithMemoryAccessLoad-Yoshi.pdf

15:15..

- **T. Shigeoka (Hitachi)**

  - **How to use "tracepoints"**
    - kernel 2.6.28から使えるtracepointsをフックしてCPU時間の測定やリモートからのモニタリングを行う方法について紹介いたします。
- Media:CELFJAM20100903-shigeoka-How-to-use-tracepoints.pdf

- Video

16:00..

- **H. Munakata (Renesas)**
    - **LinuxCon Japan session preview**
        - Unification of embedded CPU variants
            - Linux kernel の vender tree 問題の解析と提言

17:30pm..

- ***Calling for your session proposals***
- Anticipating the pre-presentation to LinuxCon Japan
    - LInuxCon Japanでセッションをもたれる方は是非その紹介を！
- Session proposal how-to. / 提案の方法
- Please be noted above time table is just a guideline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

- Tweets in Japanese. http://togetter.com/li/47157

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWikiに残してください。

English Translation Volunteer

- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

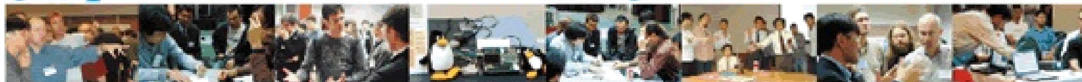
Categories:

- Japan Technical Jamboree
- Events

> From: eLinux.org

Japan Technical Jamboree 35

CE Linux Forum
Japan Technical Jamboree



Date: December 10th / 日付: 12月10日（金）
At Nakano Sunplaza / 於、中野サンプラザ

Contents

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Linux Forum. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
  - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly without any complicated expressions. Most Japanese developers are capable to understand plain English.

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。
  - Japan Technical Jamboree 34

Date and venue... / 日付・場所...

	• Date **December 10th, 2010** ◦ **Starting at 10 am** • At ***Nakano Sunplaza*** / 会場 中野サンプラザ ◦ Nakano Sunplaza is located just close to **Nakano** station (JR/Tokyo Metoro). ◦ http://www.sunplaza.jp/ (Japanese) • Admission: ***Free of charge*** / 参加費用: 無料 • Not limited for CELF members. / CELF会員以外も参加・セッション持ち込み共に可能 • Coordinators / 世話役 (Your inquiries in English welcome) ◦ Hisao Munakata / 宗像尚郎 (munakata_dot_hisao(a)renesas_dot_com) ◦ Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

Registration / 参加登録

- Registration Page / 参加登録
  - Registration is only to forecast the approximate number of the attendees. No confirmation mail will be delivered. It is not mandatory.
  - 参加登録は大まかな参加人数把握のためにお願いしています。入力をされても確認メールは発信されません。参加登録をしないで出席していただいても構いません。

Main Topics

- It is just before the LinuxCon Japan!

LinuxCon Japan直前です！

Agenda / 進行

Agenda

Time

Title and presenter

Notes

Presentation Materials

10:30..

- Opening / 連絡事項
- **CE Linux Forum to go with The Linux Foundation (S. Ueda, CELF Marketing Gp. Chair)**

11:00..

- **Event report (Kernel Summit and more) (H. Munakata, Renesas)**

12:00..

Lunch

1:00pm..

Connected from USA

- **Tim Bird**
  - Latest Community topics update
- Join jamboree from Germany through Internet
  - (English / 英語のセッションです)
- Media:Status2-of-embedded-Linux-2010-12-Jamboree35.ppt

- **Tim Bird**
  - Latest CELF development project update
    - Including brief introduction of current Open Project Proposals
- Join jamboree from Germany through Internet
  - (English / 英語のセッションです)
- See presentation above.

2:00..

- **Init of Android (T. Kobayashi, KMC)**
  - with demo using a new dual-core Cortex-A9 board

slide in English

article in Japanese

2:30..

- **Statistics tells something about Linux Community! (Tentative / T. Shibata, NEC)**

3:15..

- **An impression of Linux embedded gadget (Tentative / T. Shibata)**

3:45..

- **It's my idea! kexec-tools for busybox (Simon Horman)**

4:30pm..

- ***Calling for your session proposals***
- Session proposal how-to. / 提案の方法
- Please be noted above time table is just a guideline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

- Tweets in Japanese http://togetter.com/li/78966

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWikiに残してください。

English Translation Volunteer

- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

Categories:

- Japan Technical Jamboree
- Events

> From: eLinux.org

Japan Technical Jamboree 36

震災で被災された方に謹んでお見舞い申し上げます。

Please be noted that this Jamboree was cancelled.

3月**18**日の日本テクニカルジャンボリーは中止します。震災の影響で電源供給や公共交通機関に予期できない状況が発生する可能性が出たためです。

The Linux Foundation CE Workgroup
Japan Technical Jamboree



Date: March 18th / 日付: 3月18日（金）
At Nakano Sunplaza / 於、中野サンプラザ

Contents

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Linux Forum. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
  - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly without any complicated expressions. Most Japanese developers are capable to understand plain English.

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。
    - Japan Technical Jamboree 35

Date and venue... / 日付・場所...

<table>
<tr>
<td></td>
<td>

- Date **March 18th, 2011**
    - **Starting at 10 am**
- At ***Nakano Sunplaza*** / 会場 中野サンプラザ
    - Nakano Sunplaza is located just close to **Nakano** station (JR/Tokyo Metoro).
    - http://www.sunplaza.jp/ (Japanese)
- Admission: ***Free of charge*** / 参加費用: 無料
- Not limited for CELF members. / CELF会員以外も参加・セッション持ち込み共に可能
- Coordinators / 世話役 (Your inquiries in English welcome)
    - Hisao Munakata / 宗像尚郎 (munakata_dot_hisao(a)renesas_dot_com)
    - Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

</td>
</tr>
</table>

Registration / 参加登録

- No registration required / 参加登録は要りません。

Main Topics

- It is just before the ELC!

ELCまであと一ヶ月のタイミングです！

Agenda / 進行

Agenda

Time

Title and presenter

Notes

Presentation Materials

10:00..

- Opening / 連絡事項
- **CE Linux Forum to go with The Linux Foundation (S. Ueda, CELF Marketing Gp. Chair)**

10:30am..

Connected from USA

- **Tim Bird**

  - Latest Community topics update
- (English / 英語のセッションです)

- **Tim Bird**

  - Latest CELF development project update
    - Including brief introduction of current Open Project Proposals
- (English / 英語のセッションです)

12:00..

Lunch

1:00pm..

- **Kazuhito Narita (Sony)**
  - Tell me! How to detect heap overwriting using watchpoint?
    - Kernel 2.6.37でARMのウォッチポイントレジスタがサポートされた。これを解析が困難なメモリ上書きのバグの調査に使いたい。静的に確保されたメモリではウォッチポイントで上書きが検出できるが、動的に確保されたヒープではプログラムを起動するたびにアドレスが変わってしまうため、ウォッチポイントの設定ができない。なにか、うまい方法はないか？

1:30pm..

- **Simon Horman**
  - Adding support to Linux to boot the AP4EB and Makerel boards from the MMCIF and SDHI hardware blocks

2:00pm..

- **Masumi Meguro**
  - (TITLE to be added)
    - 数理証明（セパレーション・ロジック）を採用した、新たなコンセプトによるMonoidics静的解析ソリューション"Infer"のご紹介。メモリ操作に重点を置いた解析を行い、ソフトウエア品質を証明、もしくは発見しにくいバグを補足し欠陥を識別。品質証明による開発とビジネス上のベネフィットなど。Inferアーキテクチャと実際のワークフローイメージ。導入事例。利用形態など

2:30pm..

- ***Calling for your session proposals***

- Session proposal how-to. / 提案の方法

- Please be noted above time table is just a guideline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWiki

に残してください。

English Translation Volunteer

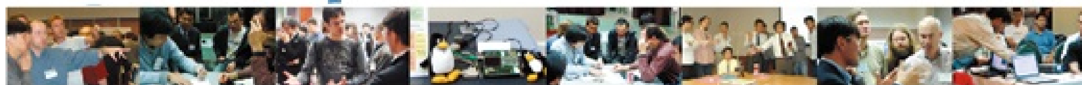
- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

Categories:

- Japan Technical Jamboree
- Events

Japan Technical Jamboree 37

The Linux Foundation CE Workgroup
Japan Technical Jamboree



Date: May 20th / 日付: 5月20日（金）
At Nakano Sunplaza / 於、中野サンプラザ

Contents

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Linux Forum. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
  - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly without any complicated expressions. Most Japanese developers are capable to understand plain English.

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。
  - Japan Technical Jamboree 35
  - Note: Japan Technical Jamboree 36 was canceled due to earthquake.

Date and venue... / 日付・場所...

	• Date **May 20th, 2011** ◦ **Starting at 10 am** • At ***Nakano Sunplaza*** / 会場 中野サンプラザ ◦ Nakano Sunplaza is located just close to **Nakano** station (JR/Tokyo Metoro). ◦ http://www.sunplaza.jp/ (Japanese) • Admission: ***Free of charge*** / 参加費用: 無料 • Not limited for CELF members. / CELF会員以外も参加・セッション持ち込み共に可能 • Coordinators / 世話役 (Your inquiries in English welcome) ◦ Hisao Munakata / 宗像尚郎 (hisao_dot_munakata_dot_vt(a)renesas_dot_com) ◦ Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

Registration / 参加登録

- No registration required / 参加登録は要りません。

Main Topics

- It is just before the LinuxCon Japan!

LinuxCon Japanまであと二週間のタイミングです！

Agenda / 進行

Agenda

Time

Title and presenter

Notes

Presentation Materials

10:00..

- Opening / 連絡事項
- **CE Linux Forum to go with The Linux Foundation (S. Ueda, CELF Marketing Gp. Chair)**

10:30am..

Connected from USA

- **Tim Bird**
  - Latest Community topics update
- (In English / 英語のセッションです)

- File:Status-of-embedded-Lnux-2011-05-J37.ppt

11:15am..

- **Tim Bird**
    - Latest CE WG project updates
        - Including brief review of current sponsored projects
- (In English / 英語のセッションです)
- File:CEWG-update-2011-05-Jamboree37.ppt

12:00..

Lunch

1:00pm..

- **Damian Hobson-Garcia (iGel)**
    - Integrating a Hardware Video Codec into Android Stagefright using OpenMAX IL
        - The presentation covers, by means of a case study, the integration of hardware video codecs into the Android Stagefright framework. The presentation discusses the basics of the OpenMAX Integration Layer (IL) API and how it is used to interconnect audio and video codecs. The method used to integrate the component into the Android Stagefright media server framework is explained. Also, some of the methods and technologies used to optimize the performance of the system, including tiled/raster conversion and hardware rendering are detailed.
- (In English / 英語のセッションです)
- http://events.linuxfoundation.org/events/embedded-linux-conference/garcia
- File:Jtl37-omxil.pdf

1:45pm..

- **Katsuya Matsubara (iGel)**
    - Integrating a Hardware Video Codec into Android Stagefright using OpenMAX IL
        - Demonstration

2:00pm..

- **Okamoto Kouta (Toshiba)**
    - Latency Bottleneck Analysis by Ftrace
        - Ftraceを利用して、実際にボトルネックとなっている箇所の特定を行いました。
- File:Latency Bottleneck Analysis by Ftrace-20110520.pdf

2:45pm..

- **Mitsuhiro Kimura (Toshiba)**
    - Evaluation of TIPC
        - Linux に組み込まれている通信プロトコル TIPC について、組込環境で TCP/IP と比較評価を行いました。
- File:TIPC CELF Japan TJ37.pdf

3:30pm..

- **Simon Horman**
    - Adding support to Linux to boot the AP4EB and Makerel boards from the MMCIF and SDHI hardware blocks

4:00pm..

- **Shinsuke Kato (Panasonic)**
    - Embedded Linux Conference / Android Builders Summit Report
- Presentation(Japanese) File:ELC ABS 2011 Report Jamboree.ppt (最終版に差し替えました 5/20 17時)

5:00pm..

- **Tetsuyuki Kobayashi (KMC)**
  - Android is NOT just 'Java on Linux'
- http://kobablog.wordpress.com/2011/05/22/android-is-not-just-java-on-linux/

5:30pm..

- ***Calling for your session proposals***
- Session proposal how-to. / 提案の方法
- Please be noted above time table is just a guideline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

- Tweets in Japanese.http://togetter.com/li/138027

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWikiに残してください。

English Translation Volunteer

- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

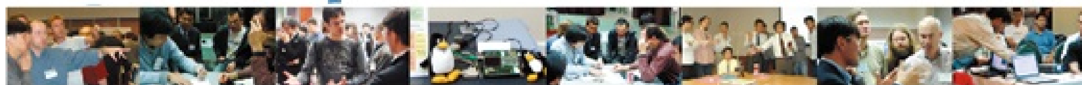
Categories:

- Japan Technical Jamboree
- Events

> From: eLinux.org

Japan Technical Jamboree 38

The Linux Foundation CE Workgroup
Japan Technical Jamboree



Date: September 30th / 日付: 9月30日（金）
At Nakano Sunplaza / 於、中野サンプラザ

SPECIAL NOTICE

- メーリングリスト管理システムが停止しているため、開催案内が出ていませんが、開催予定に変更はありません。

Contents

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Linux Forum. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
  - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly without any complicated expressions. Most Japanese developers are capable to understand plain English.

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。
  - Japan Technical Jamboree 37

Date and venue... / 日付・場所...

<table>
<tr>
<td></td>
<td>

- Date **September 30th, 2011**
  - **Starting at 10 am**
- At ***Nakano Sunplaza*** / 会場 中野サンプラザ
  - Nakano Sunplaza is located just close to **Nakano** station (JR/Tokyo Metoro).
  - http://www.sunplaza.jp/ (Japanese)
- Admission: ***Free of charge*** / 参加費用: 無料
- Not limited for CELF members. / CELF会員以外も参加・セッション持ち込み共に可能
- Coordinators / 世話役 (Your inquiries in English welcome)
  - Hisao Munakata / 宗像尚郎 (hisao_dot_munakata_dot_vt(a)renesas_dot_com)
  - Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

</td>
</tr>
</table>

Registration / 参加登録

- No registration required / 参加登録は要りません。

Main Topics

- It is just before the ELC Europe!

Embedded Lunux Conference Europeまであと一ヶ月のタイミングです！

Agenda / 進行

Agenda

Time

Title and presenter

Notes

Presentation Materials

10:00..

- Opening / 連絡事項
- **CE Linux Forum to go with The Linux Foundation (S. Ueda, CELF Marketing Gp. Chair)**

10:30am..

Connected from USA

- **Tim Bird**
  - Latest Community topics update
  - Latest CE WG project updates

- Media:Status-of-embedded-Linux-2011-09-Jamboree38.ppt
- (In English / 英語のセッションです)

12:00..

Lunch

1:00pm..

File system performance comparison for recording functions

- **N. Okabayashi**
- File:CELF 2011 09 30 in English.pdf

1:30pm..

Discussion About Long Term Support version of Linux for Embedded system

- **H. Munakata, T. Shibata, S. Ueda**

3:30pm..

- ***Calling for your session proposals***
- Session proposal how-to. / 提案の方法
- Please be noted above time table is just a guideline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

- Tweets in Japanese.http://togetter.com/li/138027

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWikiに残してください。

English Translation Volunteer

- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

Categories:

- Japan Technical Jamboree
- Events

From: eLinux.org

Japan Technical Jamboree 39

The Linux Foundation CE Workgroup
Japan Technical Jamboree



Date: December 9th/ 日付: 12月9日（金）
At Nakano Sunplaza / 於、中野サンプラザ

Contents

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Linux Forum. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
  - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly without any complicated expressions. Most Japanese developers are capable to understand plain English.

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。
  - Japan Technical Jamboree 38

Date and venue... / 日付・場所...

<table>
<tr>
<td></td>
<td>

- Date **December 9th, 2011**
  - **Starting at 10 am**
- At ***Nakano Sunplaza*** / 会場 中野サンプラザ
  - Nakano Sunplaza is located just close to **Nakano** station (JR/Tokyo Metoro).
  - http://www.sunplaza.jp/ (Japanese)
- Admission: ***Free of charge*** / 参加費用: 無料
- Not limited for CELF members. / CELF会員以外も参加・セッション持ち込み共に可能
- Coordinators / 世話役 (Your inquiries in English welcome)
  - Hisao Munakata / 宗像尚郎 (hisao_dot_munakata_dot_vt(a)renesas_dot_com)
  - Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

</td>
</tr>
</table>

Registration / 参加登録

- No registration required / 参加登録は要りません。

Main Topics

- It is just after the ELC Europe!

Embedded Lunux Conference まであと二ヶ月のタイミングです！

Agenda / 進行

Agenda

Time	Title and presenter	Notes Presentation Materials
10:00..	• Opening / 連絡事項 • **CE Linux Forum to go with The Linux Foundation (S. Ueda, CELF Marketing Gp. Chair)**	
10:30am..	**Connected from USA** • **Tim Bird** 　◦ Latest Community topics update 　◦ Latest CE WG project updates	• File:Status-of-Embedded-Linux-2011-12-JJ39.pdf (In English / 英語のセッションです)
12:00..	Lunch	
1:00pm..	**Repeat! ELC-E 2011: Contributing to the Community? Does your manager support you?** • **S. Ueda**	• File:EEF2011 Invitation.pdf
2:00pm..	**Repeat! ELC-E 2011: How Linux-RT Works** • **Frank Rowand** • **Connected from USA**	• File:How linux preempt rt works 111207 1100.pdf • (In English / 英語のセッションです)
2:45pm..	**A topics about Android Compatibility Definition** • **Shinsuke Kato (Panasonic)**	• File:Android CDD Jamboree 39.pdf
3:30pm..	**Status of LTSI project** • **Hisao Munakata**	
4:30pm..	**Evaluation of IEEE802.3az on Linux Ethernet Driver** • **Hiroo MATSUMOTO, FUJITSU COMPUTER TECHNOLOGIES LIMITED**	• File:Evaluation of IEEE802.3az on Linux Ethernet Driver.pdf
4:50pm..	**Evaluation of IEEE1588 on Linux Ethernet Driver** • **Akio MORI, FUJITSU COMPUTER TECHNOLOGIES LIMITED**	• File:Evaluation of IEEE1588 on Linux Ethernet Driver.pdf
5:10pm..	• ***Calling for your session proposals***	• Session proposal how-to. / 提案の方法

- Please be noted above time table is just a guideline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

- Tweets in Japanese.http://togetter.com/li/138027

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWikiに残してください。

English Translation Volunteer

- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

Categories:

- Japan Technical Jamboree
- Events

> From: eLinux.org

Japan Technical Jamboree 40

The Linux Foundation CE Workgroup
Japan Technical Jamboree



Date: March 23rd/ 日付: 3月23日（金）

- At Nakano Sunplaza / 於、中野サンプラザ
- Please join "celinux-dev" mailinglist to get the latest announcement.
- "celinux-dev" メーリングリストにこのイベント関係のアナウンスが流れます。ぜひ参加してください。
  - How to join.

Contents

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Workgroup of the Linux Foundation. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
  - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly without any complicated expressions. Most Japanese developers are capable to understand plain English.

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。

- Japan Technical Jamboree 39

Date and venue... / 日付・場所...

<table>
<tr>
<td></td>
<td>

- Date **March 23rd, 2012**
  - **Starting at 10 am**
- At ***Nakano Sunplaza*** / 会場 中野サンプラザ
  - Nakano Sunplaza is located just close to **Nakano** station (JR/Tokyo Metoro).
  - http://www.sunplaza.jp/ (Japanese)
- Admission: ***Free of charge*** / 参加費用: 無料
- Not limited for CELF members. / CELF会員以外も参加・セッション持ち込み共に可能
- Coordinators / 世話役 (Your inquiries in English welcome)
  - Hisao Munakata / 宗像尚郎 (hisao_dot_munakata_dot_vt(a)renesas_dot_com)
  - Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

</td>
</tr>
</table>

Registration / 参加登録

- No registration required / 参加登録は要りません。

Main Topics

- It is just after the ELC!

Embedded Lunux Conference の後をうけた開催です。

Agenda / 進行

Agenda

Time	Title and presenter	Notes Presentation Materials
10:00..	• Opening / 連絡事項	• File:SelfIntro40.ppt
10:30am..	**Embedded Linux status update** • **Tim Bird** (Connect from USA) ◦ Latest Community topics update - quick update. ◦ Latest CE WG project updates	(In English / 英語のセッションです) Media:Status-update-2012-03-Jamboree40.pdf
11:00am..	**Embedded-Appropriate Crash Handling in Linux (from ELC 2012)** • **Tim Bird, Sony Network Entertainment**	(In English / 英語のセッションです) Media:Embedded-crash-handling-2012-03.pdf

	◦ See ELC abstract	
12:00..	Lunch	
1:00pm..	**(The latest status of LTSI project)** • **H. Munakata / T. Shibata** ◦ (TBD)	
1:30pm..	**(from Android Builders Summit 2012)** • **T. Kobayashi (KMC)** ◦ "ADB(Android Debug Bridge):How it works"	link to slide
2:30pm..	**(from Android Builders Summit 2012)** • **S. Kato (Panasonic)** ◦ (45分)"Android Compatibility" and the Way to Check It Android互換性の話をします。12月のJamboree #39の内容をUpdateし、社内の取り組み事例を追加しています！ • ◦ (15分)ELC参加レポート 今回はビデオアーカイブもあるのでショートバージョンです。ABS Videos, ELC Videos	• link to slide（"Android Compatibility" and the Way to Check It） • File:Android CDD Jamboree 39.pdf 前回のJamboreeの資料(Android-2.3.3のCDDの概要説明) • Media:ABS_ELC_2012_Report_Jamboree_v1.1.ppt link to slide(ABS,ELC report) • File:Report120323abselc2012.pdf ABS/ELC2012 Report, presented by Matsubara@igel
3:30pm..	**Ineffective and Effective Ways To Find Out Latency Bottlenecks With Ftrace (from ELC 2012)** • **Yoshitake Kobayashi (TOSHIBA)**	link to slide
4:15pm..	**Using Linux DVFS to control a switcher(hypervisor) between ARM Cortex A15 & A7** • **Sylvain Bayon de Noyer (Synopsys)** ◦ ARM is coming with a new architecture so called big.LITTLE with a small (low power) core Cortex A7, and a bigger one Cortex A15. ◦ This architecture has the capability to switch the execution between	• File:DVFS for ARM CortexA15 A7.pdf

	both cores, balancing the performance vs. power. ◦ We have been experimenting with this capability specifically with Linux and also android. ◦ This session covers few explanation of the this technology, our experiments and results.	
5:00pm..	**Introduction to Linaro** • **Akira Tsukamoto, Linaro**	
5:15pm..	**Linaro Connect Q1.12 Report** • **Teppei Asaba, Fujitsu Computer Technologies Ltd.**	
5:30pm..	**Chromium OS Build system** • **Mitsuhiro Kimura (TOSHIBA)** ◦ Talking about Chromium OS build steps from making chroot environment until writing the image to a USB memory.	File:CELF2012 kimura 20120322 2.pdf
6:00pm..	• ***Calling for your session proposals***	• Session proposal how-to. / 提案の方法

- Please be noted above time table is just a guideline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWikiに残してください。

English Translation Volunteer

- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

Categories:

- Japan Technical Jamboree
- Events

From: eLinux.org

Japan Technical Jamboree 41

The Linux Foundation CE Workgroup

Japan Technical Jamboree



Date: June 21st/ 日付: 6月21日（木）

- At Nakano Sunplaza / 於、中野サンプラザ
- Please join "celinux-dev" mailinglist to get the latest announcement.
- "celinux-dev" メーリングリストにこのイベント関係のアナウンスが流れます。ぜひ参加してください。
    - How to join.

Contents

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Workgroup of the Linux Foundation. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
    - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly without any complicated expressions. Most Japanese developers are capable to understand plain English.

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。

- Japan Technical Jamboree 40

Date and venue... / 日付・場所...

<table>
<tr>
<td></td>
<td>

- Date **June 21st, 2012**
  - **Starting at 10 am**
- At ***Nakano Sunplaza / 8F Training Room 1*** / 会場 中野サンプラザ / **8**階・研修室１
  - Nakano Sunplaza is located just close to **Nakano** station (JR/Tokyo Metoro).
  - http://www.sunplaza.jp/ (Japanese)
- Admission: ***Free of charge*** / 参加費用: 無料
- Not limited for CELF members. / CELF会員以外も参加・セッション持ち込み共に可能
- Coordinators / 世話役 (Your inquiries in English welcome)
  - Hisao Munakata / 宗像尚郎 (hisao_dot_munakata_dot_vt(a)renesas_dot_com)
  - Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

</td>
</tr>
</table>

Registration / 参加登録

- No registration required / 参加登録は要りません。

Main Topics

- It is just after the LinuxCon Japan!

LinuxCon Japan の後をうけた開催です。

- Major progress will be expected about LTSI project.

LTSIプロジェクトに大きな進展がある予定です。

Agenda / 進行

Agenda

Time	Title and presenter	Notes Presentation Materials
10:00am..	**Embedded Linux status update** • **Tim Bird** (Connect from USA) ◦ Latest Community topics update - quick update. ◦ Latest CE WG project updates	(In English / 英語のセッションです) File:Status-of-Embedded-Linux-20 JJ41.ppt
10:45am..	• Opening / 連絡事項	• File:SelfIntro40.ppt
11:00am..	**Lightning Talk Time**	

11:30am..	Lunch	
12:30pm..	**The latest status of LTSI project** • **H. Munakata (T. Shibata)** **My experience to donate patches to LTSI** • **Satoshi Uchino (Toshiba)**	• File:LTSI-submission-uchino.pdf
1:15pm..	**(from LinuxCon Japan 2012) Applying Clang Static Analyzer to Linux Kernel** • **Hiroo Matsumoto, Fujitsu Computer Technologies Ltd.**	https://events.linuxfoundation.org/images/stories/pdf/lcjp2012_matsumo
2:00pm..	**Status of Linux 3.x Realtime and Changes From 2.6** • **Frank Rowand** (Connect from USA)	(In English / 英語のセッションです) https://events.linuxfoundation.org/events/linuxcon-japan/rowand
2:45pm..	**(from LinuxCon Japan 2012) Running uClinux on ARM Cortex-M3 Platform** • **Sun Wei, Fujitsu Computer Technologies Ltd.**	https://events.linuxfoundation.org/images/stories/pdf/lcjp2012_wei.pdf
3:30pm..	**Research and evaluation of SCHED_DEADLINE** • **Masahiro Yamada, TOSHIBA** ◦ タスクのバジェット管理によるEDFスケジューリングを行うSCHED_DEADLINEについて調査，分析を行いま した．	• File:Sched deadline.pdf
4:00pm..	Tea break	
4:15pm..	**Introduction of KZM-A9-GT (and may be more)** • **Tetsuyuki Kobayashi, KMC** • Latest kernel(3.5-rc3) and LTSI3.0 kernel run on it.	• http://www.kmckk.co.jp/kzma9-gt/index.html
	LinuxCon Japan 2012 report • **Shinsuke Kato (Panasonic)** ◦ みなさま参加された	• (PDF) File:LinuxConJapan2012 report Jamboree41.pdf

5:00pm..	方が多いと思いますが、LinuxCon Japanの参加レポートをしようと思います(15分～20分くらいで)	• (PPT) File:LinuxConJapan2012 report Jamboree41.ppt
5:30pm..	**Updates of Linaro Connect Q2.12 May. 28 – Jun. 1, 2012 at Hong Kong** • **Akira Tsukamoto (Linaro)**	https://events.linuxfoundation.org/images/stories/pdf/lcjp2012_tsukamo
6:00pm..	**Japan OSS Promotion Forum / Embedded System Sub-Group Meeting** • **S. Ueda** ◦ It is the work group meeting though open to everyone to attend. ◦ 日本OSS推進フォーラム、組込みシステム部会のミーティングですが、どなたでも参加可能です。	• Session proposal how-to. / 提案の方法

- Please be noted above time table is just a guideline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

- togetter

http://togetter.com/li/324703

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWikiに残してください。

English Translation Volunteer

- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

Categories:

- Japan Technical Jamboree
- Events

> From: eLinux.org

Japan Technical Jamboree 42

The Linux Foundation CE Workgroup
Japan Technical Jamboree



Date: September 20th/ 日付: 9月20日（木）

- At Nakano Sunplaza / 於、中野サンプラザ
- Please join "celinux-dev" mailinglist to get the latest announcement.
- "celinux-dev" メーリングリストにこのイベント関係のアナウンスが流れます。ぜひ参加してください。
    - How to join.

Contents

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Workgroup of the Linux Foundation. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
    - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly without any complicated expressions. Most Japanese developers are capable to understand plain English.

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。
  - Japan Technical Jamboree 41

Date and venue... / 日付・場所...

	• Date **September 20th, 2012** ◦ **Starting at 10 am** • At ***Nakano Sunplaza / 8F Training Room 2*** / 会場 中野サンプラザ / **8**階・研修室２ ◦ Nakano Sunplaza is located just close to **Nakano** station (JR/Tokyo Metoro). ◦ http://www.sunplaza.jp/ (Japanese) • Admission: ***Free of charge*** / 参加費用: 無料 • Not limited for CELF members. / CELF会員以外も参加・セッション持ち込み共に可能 • Coordinators / 世話役 (Your inquiries in English welcome) ◦ Hisao Munakata / 宗像尚郎 (hisao_dot_munakata_dot_vt(a)renesas_dot_com) ◦ Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

Registration / 参加登録

- No registration required / 参加登録は要りません。

Main Topics

- It is just after the Kernel Summit and final Jamboree before ELC Europe.

Agenda / 進行

Agenda

Time	Title and presenter	Notes Presentation Materials
10:00..	• Opening / 連絡事項	
10:15am..	**Embedded Linux status update** • **Tim Bird** (Connect from USA) ◦ Latest Community topics update - quick update. ◦ Latest CE WG project updates	(In English / 英語のセッションです) • File:Status of Embedded Linux-2012-09-JJ42.ppt • http://youtu.be/tVkmfnkfE6g
12:00..	Lunch	
1:00pm..	**Status of Linux 3.x Realtime and Changes From 2.6**	(In English / 英語のセッションです) • https://events.linuxfoundation.org/events/linuxcon-japan/rowand

	• **Frank Rowand** (Connect from USA)	• File:Status of real time.pdf • http://youtu.be/g0YfhYj37N0
2:00pm..	**The latest status of LTSI project** • **S. Ueda** (On behalf of Shibata san and Munakata san) • The uploaded video is presentation **performed by Munakata san by himself**.	• File:Ltsi2012lcna.pdf • http://youtu.be/5FKnph8Gpro • http://youtu.be/z9yD0Kmu9oU
2:30pm..	コミュニティーデビューの体験をゆる～く • How to post patch 「初めてのパッチ投稿(体験談)」 • **Tetsuyuki Kobayashi**	• http://www.slideshare.net/tetsu.koba/patch101 • http://youtu.be/ZBS3P0J4m-8
3:15pm..	**Evaluation of Flash Filesystems Update** • SFTL (Simple Flash Translation Layer) • **K. Uwatoko**	• File:Celf sftl.pdf • http://youtu.be/ifP5JAHO_v4
4:00pm..	**Developing Embedded Linux by Poky** • **K.Hayashi**, TOSHIBA • ビルドシステムPokyを使用して、オリジナルのLinux環境を構築する方法についてご説明します	• File:Celfjtj42 poky.pdf • http://youtu.be/2HNp-jFbz2o
4:45pm..	**A little complicated issue / Legal Compliance** • Introduction of Open Compliance Summit • **S. Ueda**	• https://events.linuxfoundation.jp/events/open-compliance-summit-asia-edition
6:00pm	***Estimated*** Closing Time	• Note: The room is booked until 9:00pm.

- Please be noted above time table is just a guideline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

Collection of tweet / Togetter

- Collection of tweets during the event is here:
  - http://togetter.com/li/376617

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWikiに残してください。

English Translation Volunteer

- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

Categories:

- Japan Technical Jamboree
- Events

> From: eLinux.org

Japan Technical Jamboree 43

The Linux Foundation CE Workgroup

Japan Technical Jamboree



Date: December 7th/ 日付: 12月7日（金）

- At Nakano Sunplaza / 於、中野サンプラザ
- Please join "celinux-dev" mailinglist to get the latest announcement.
- "celinux-dev" メーリングリストにこのイベント関係のアナウンスが流れます。ぜひ参加してください。
  - How to join.

Contents

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Workgroup of the Linux Foundation. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
  - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly without any complicated expressions. Most Japanese developers are capable to understand plain English.

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。

- Japan Technical Jamboree 42

Date and venue... / 日付・場所...

<table>
<tr>
<td></td>
<td>

- Date **December 7th, 2012**
  - **Starting at 10 am**
- At ***Nakano Sunplaza / 8F Training Room 2*** / 会場 中野サンプラザ **/ 8**階・研修室２
  - Nakano Sunplaza is located just close to **Nakano** station (JR/Tokyo Metoro).
  - http://www.sunplaza.jp/ (Japanese)
- Admission: ***Free of charge*** / 参加費用: 無料
- Not limited for CELF members. / CELF会員以外も参加・セッション持ち込み共に可能
- Coordinators / 世話役 (Your inquiries in English welcome)
  - Hisao Munakata / 宗像尚郎 (hisao_dot_munakata_dot_vt(a)renesas_dot_com)
  - Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

</td>
</tr>
</table>

Registration / 参加登録

- No registration required / 参加登録は要りません。

Main Topics

- It is just after the Kernel Summit and final Jamboree before ELC Europe.

Agenda / 進行

Agenda

Time	Title and presenter	Notes Presentation Materials
10:00..	• Opening / 連絡事項	
10:30am..	**Embedded Linux status update and The latest status of LTSI project** • **H. Munakata for Tim Bird** ◦ Latest Community topics update - quick update. ◦ Latest CE WG project updates ◦ Refered URI • ◦ LWN kernel index page(see Release section) ◦ kernelnewbies kernel change log ◦ ELCE2012 sides	• File:Status-of-embedded Linux-2012-12-JJ42.pdf • File:Introduction to Ne10.pdf • Video

	◦ collection ◦ Linaro published slides ◦ Korea Linux Forum 2012 ◦ Project Ne10: An Open Optimized Software Library Project for the ARM ◦ LTSI project ML achives ◦ Yocto project doccumentation	
11:30..	Lunch	
1:00pm..	組み込みエンジニアのための **Linux**入門　仮想メモリ編 • **Tetsuyuki Kobayashi**	• http://www.slideshare.net/tetsu.koba/basic-of-virtual-memory-of-linux • Video
1:45pm..	**Improvement of Scheduling Granularity for Deadline Scheduler (From ELC-E2012)** • **Y. Kobayashi**	• Slide • Video
2:30pm..	**Integrating video processing hardware into GStreamer:** • a case study of the Renesas R-CarE1 platform • **K. Matsubara**	File:Gstreamer121207cewgjamboree43.pdf • Video
3:15pm..	**Linux Contiguous Memory Allocator support** • The presentation will discuss the functionality and use cases of the Linux Contiguous Memory Allocator, and how it can be used by both kernel and user space drivers. • **Damian Hobson-Garcia**	File:LinuxCMA-cewg43.pdf • Video
4:00pm..	**A Special Talk** • **S. Ueda**	• File:A SpecialTalk.pdf • Video
4:45pm..	• ***Calling for your session proposals***	• Session proposal how-to. / 提案の方法

- Please be noted above time table is just a guideline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

- togetter

http://togetter.com/li/418981

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWikiに残してください。

English Translation Volunteer

- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

Categories:

- Japan Technical Jamboree
- Events

> From: eLinux.org

Japan Technical Jamboree 44

The Linux Foundation CE Workgroup
Japan Technical Jamboree



Date: March 8th/ 日付: 3月8日（金）

- At Nakano Sunplaza / 於、中野サンプラザ
- Please join "celinux-dev" mailinglist to get the latest announcement.
- "celinux-dev" メーリングリストにこのイベント関係のアナウンスが流れます。ぜひ参加してください。
  - How to join.

Contents

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Workgroup of the Linux Foundation. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
  - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly without any complicated expressions. Most Japanese developers are capable to understand plain English.

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。
    - Japan Technical Jamboree 43

Hash tag on Twitter

- **#CELFJP**

Date and venue... / 日付・場所...

<table>
<tr>
<td></td>
<td>

Date March 8th, 2013
Starting at 10 am
At Nakano Sunplaza / 8F Training Room 2 / 会場 中野サンプラザ / 8階・研修室２
Nakano Sunplaza is located just close to Nakano station (JR/Tokyo Metoro).
http://www.sunplaza.jp/ (Japanese)
Admission: Free of charge / 参加費用: 無料
Not limited for CELF members. / CELF会員以外も参加・セッション持ち込み共に可能
Coordinators / 世話役 (Your inquiries in English welcome)
Hisao Munakata / 宗像尚郎 (hisao_dot_munakata_dot_vt(a)renesas_dot_com)
Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

</td>
</tr>
</table>

Registration / 参加登録

- No registration required / 参加登録は要りません。

Main Topics

- It is just after the Kernel Summit and final Jamboree before ELC Europe.

Agenda / 進行

Agenda

<table>
<tr><th>Time</th><th>Title and presenter</th><th>Notes
Presentation Materials</th></tr>
<tr><td>10:00..</td><td>Opening / 連絡事項</td><td></td></tr>
<tr><td>10:15am..</td><td>Embedded Linux status update
Tim Bird (Connect from USA)
Latest Community topics update - quick update.
Latest CE WG project updates</td><td>(In English / 英語のセッションで
Media:Status-of-Embedded-
Media:Status-of-Embedded-
Video</td></tr>
<tr><td>11:30..</td><td>Lunch</td><td></td></tr>
<tr><td>12:30pm..</td><td>Evalution of Linux Container (LXC) on Embedded Linux
Yuki Machida, FUJITSU COMPUTER TECHNOLOGIES</td><td>Media:evalution_of_linux_cc
Video</td></tr>
</table>

	LIMITED	• Video
1:30pm..	**barebox** with GUI and the up comming Application support • **Jean-Christophe PLAGNIOL-VILLARD**	(In English / 英語のセッションで • Connecting from Hong Kong • http://www.jcrosoft.com/ • Media:barebox.pdf • Video
2:00pm..	**The latest status of LTSI project** • **Hisao Munakata**	• Media:LTSIwithYocto.pdf • Video
2:30pm..	**Tips of Malloc & Free** • **Tetsuyuki Kobayashi**	http://d.hatena.ne.jp/embedded/2 • Video
3:30pm..	**Applying the Linux Cgroups Mechanism to AV appliances** • **Yukinori Endo / Mitsubishi Electric Corp.**	• Media:Jamboree44_Cgroup • Video
4:00pm..	**Android Builders Summit 2013 / Embedded Linux Conference 2013 Report** • **Kazuomi Kato, Panasonic** • **Shinsuke Kato, Panasonic** ◦ ABSとELCの参加レポートを予定しています。セッションは終わりの方でOKです。個別の話題を前に追記下さい。 • Togetter ◦ Android Builders Summit 2013 #abs2013 http://togetter.com/li/458582 ◦ Yocto Project Developer Day #yocto http://togetter.com/li/459273 ◦ Embedded Linux Conference 2013 #lfelc http://togetter.com/li/459672 • Links ◦ ABS https://events.linuxfoundation.org/events/android-builders-summit ◦ ELC https://events.linuxfoundation.org/events/embedded-linux-conference • Links after archived ◦ ABS https://events.linuxfoundation.org/archive/2013/android-builders-summit ◦ ELC https://events.linuxfoundation.org/archive/2013/embedded-linux-conference	• File:ABS ELC 2013 Report • Video
4:40pm..	**Embedded Linux Conference 2013 Closing Session** から • **Kazuomi Kato, Panasonic** • **Shinsuke Kato, Panasonic** ◦ ABSとELCのClosing Sessionの紹介。 Thank you for Tim!	
5:00pm..	• ***Calling for your session proposals***	• Session proposal how-to. / 招

- Please be noted above time table is just a guideline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

- togetter
  - http://togetter.com/li/468165

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWikiに残してください。

English Translation Volunteer

- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

Categories:

- Japan Technical Jamboree
- Events

From: eLinux.org

Japan Technical Jamboree 45

The Linux Foundation CE Workgroup

Japan Technical Jamboree



Date: June 7th/ 日付: 6月7日（金）

- At Nakano Sunplaza / 於、中野サンプラザ
- Please join "celinux-dev" mailinglist to get the latest announcement.
- "celinux-dev" メーリングリストにこのイベント関係のアナウンスが流れます。ぜひ参加してください。
  - How to join the mailing list.

Contents

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Workgroup of the Linux Foundation. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
  - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly without any complicated expressions. Most Japanese developers are capable to understand plain English.

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。
    - Japan Technical Jamboree 44

Hash tag on Twitter

- **#CELFJP**

Date and venue... / 日付・場所...

	• Date **June 7th, 2013** 　◦ **Starting at 10 am** • At ***Nakano Sunplaza / 8F Training Room 2*** / 会場 中野サンプラザ ***/ 8***階・研修室２ 　◦ Nakano Sunplaza is located just close to **Nakano** station (JR/Tokyo Metoro). 　◦ http://www.sunplaza.jp/ (Japanese) • Admission: ***Free of charge*** / 参加費用: 無料 • Not limited for CE Workgroup members. / CE Workgroup会員以外も参加・セッション持ち込み共に可能 • Coordinators / 世話役 (Your inquiries in English welcome) 　◦ Hisao Munakata / 宗像尚郎 (hisao_dot_munakata_dot_vt(a)renesas_dot_com) 　◦ Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

Registration / 参加登録

- No registration required / 参加登録は要りません。

Main Topics

- It is just after the Kernel Summit and final Jamboree before ELC Europe.

Agenda / 進行

Agenda

Time

Title and presenter

Notes

Presentation Materials

10:00..

- Opening / 連絡事項

- File:SelfIntro45.pdf

- CE Linux Forumの「伝説の仕様書」はここにあります。
    - http://tree.celinuxforum.org/CelfPubWiki/FrontPage#Requirements_and_Specifications

10:15am..

Embedded Linux status update

- **Tim Bird** (Connect from USA)
  - Latest Community topics update - quick update.
  - Latest CE WG project updates
- (In English / 英語のセッションです)

- Media:Status-of-Embedded-Linux-2013-06-JJ45.pdf
- Video

11:30..

Lunch

12:30pm..

Simple and efficient way to get the last log using MMAP

- **Tetsuyuki Kobayashi, Kyoto microcomputer**

- http://www.slideshare.net/tetsu.koba/simple-and-efficient-way-to-get-the-last-log-using-mmap

- Video

1:15pm..

Implementing Genivi IVI Layer Management With DirectFB

- **Takanari Hayama**

- http://events.linuxfoundation.org/sites/events/files/alss13_hayama.pdf

- Video

2:00pm..

One User Space Approach to big.LITTLE MP System on Real Silicon

- **Tetsuya Nakagawa, Renesas**

- Media:lcjp2013_nakagawa.pdf

- Video

2:45pm..

Linux Fast Boot

- **Eiji Kameyama, FUJITSU COMPUTER TECHNOLOGIES LIMITED**

- Media:LinuxFastBoot.pdf

- Video

3:30pm..

LTSI updates & ALS2013 my session quick review

- **Hisao Munakata (Renesas)**

- Media:LTSI-collabsummit.pdf

- Video

4:15pm..

Real-Time Task Partitioning using Cgroups

- **Akihiro Suzuki, TOSHIBA**
- Media:Real-Time_Tasks_Partitioning_using_Cgroups.pdf
- Video

5:00pm..

The short report of Automotive Linux Summit / LinuxCon Japan 2013

- **Kazuomi Kato, Panasonic**
- File:ALS LinuxCon Japan 2013 Report Jamboree.zip (zipの中身はpptです)
- Video

5:30pm..

- **'Call for session'**
- Session proposal how-to. / 提案の方法
- Please be noted above time table is just a guideline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

- togetter
  - http://togetter.com/li/514818

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWikiに残してください。

English Translation Volunteer

- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

Jamboree Scene of 45th



- Not only "Consumer Electronics" people but some "Autumotive" people joined.
- About 30 to 40 participants.

Categories:

- Japan Technical Jamboree
- Events

> From: eLinux.org

Japan Technical Jamboree 46



Date: September 13th/ 日付: 9月13日（金）

- At Nakano Sunplaza / 於、中野サンプラザ
- Please join "celinux-dev" mailinglist to get the latest announcement.
- "celinux-dev" メーリングリストにこのイベント関係のアナウンスが流れます。ぜひ参加してください。
  - How to join the mailing list.

Contents

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Workgroup of the Linux Foundation. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
  - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly without any complicated expressions. Most Japanese developers are capable to understand plain English.

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。
  - Japan Technical Jamboree 45

Hash tag on Twitter

- **#CELFJP**

Date and venue... / 日付・場所...

<table>
<tr>
<td></td>
<td>

Date September 13th, 2013
Starting at 10 am
At Nakano Sunplaza / 8F Training Room 2 / 会場 中野サンプラザ / 8階・研修室２
Nakano Sunplaza is located just close to Nakano station (JR/Tokyo Metoro).
http://www.sunplaza.jp/ (Japanese)
Admission: Free of charge / 参加費用: 無料
Not limited for CE Workgroup members. / CE Workgroup会員以外も参加・セッション持ち込み共に可能
Coordinators / 世話役 (Your inquiries in English welcome)
Hisao Munakata / 宗像尚郎 (hisao_dot_munakata_dot_vt(a)renesas_dot_com)
Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

</td>
</tr>
</table>

Registration / 参加登録

- No registration required / 参加登録は要りません。

Main Topics

- It is just after the Kernel Summit and final Jamboree before ELC Europe.

Agenda / 進行

Agenda

Time

Title and presenter

Notes

Presentation Materials

10:00..

- Opening / 連絡事項

11:00am..

LTSI project status update

- **Hisao Munakata**
- Presentation Material
  - [JP] http://www.linuxfoundation.jp/news-media/jp_lets-work-together-ltsi-310
  - [US] http://www.linuxfoundation.org/news-media/blogs/browse/2013/09/quick-guide-get-ready-ltsi-310
- Video

11:30..

Lunch

1:00am..

Embedded Linux status update

- **Tim Bird** (Connect from USA)
  - Latest Community topics update - quick update.
  - Latest CE WG project updates

(In English / 英語のセッションです)

- Media:Status-of-Embedded-Linux-2013-09-JJ46.pdf
- Video

2:15pm..

Discussion: Use of source code hosting services such as Github, Bitbucket and so on

3:30pm..

How to tune ext4 filesystem for Embedded System

- **Yuki Machida, FUJITSU COMPUTER TECHNOLOGIES LIMITED**
- Video

4:30pm..

Improving the real-time performance with CPU affinity

- **Yoshitake Kobayashi, TOSHIBA**
- Media:CEWG-TechJam46-Kobayashi.pdf
- Video

5:15pm..

- **'Call for session'**
- Session proposal how-to. / 提案の方法
- Please be noted above time table is just a guideline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

- togetter
  - http://togetter.com/li/563301

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWikiに残してください。

English Translation Volunteer

- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

Categories:

- Japan Technical Jamboree
- Events

From: eLinux.org

Japan Technical Jamboree 47



Date: December 11th/ 日付: 12月11日（水）DATE CHANGED

- At Nakano Sunplaza / 於、中野サンプラザ
- Please join "celinux-dev" mailinglist to get the latest announcement.
- "celinux-dev" メーリングリストにこのイベント関係のアナウンスが流れます。ぜひ参加してください。
  - How to join the mailing list.

 We are calling for session proposal now! / セッション提案募集中！

Session proposal how-to. / 提案の方法

Contents

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Workgroup of the Linux Foundation. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
  - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly without any complicated expressions. Most Japanese developers are capable to understand plain English.

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。
    - Japan Technical Jamboree 46

Hash tag on Twitter

- **#CELFJP**

Date and venue... / 日付・場所...

<table>
<tr>
<td></td>
<td>

- Date **December 11th, 2013** DATE CHANGED
    - **Starting at 10 am**
- At ***Nakano Sunplaza / 8F Training Room 5*** / 会場 中野サンプラザ / ***8***階・研修室５
    - Nakano Sunplaza is located just close to **Nakano** station (JR/Tokyo Metoro).
    - http://www.sunplaza.jp/ (Japanese)
- Admission: ***Free of charge*** / 参加費用: 無料
- Not limited for CE Workgroup members. / CE Workgroup会員以外も参加・セッション持ち込み共に可能
- Coordinators / 世話役 (Your inquiries in English welcome)
    - Hisao Munakata / 宗像尚郎 (hisao_dot_munakata_dot_vt(a)renesas_dot_com)
    - Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

</td>
</tr>
</table>

Registration / 参加登録

- No registration required / 参加登録は要りません。

Main Topics

- It is just after the Kernel Summit and final Jamboree before ELC Europe.

Agenda / 進行

Agenda

Time

Title and presenter

Notes

Presentation Materials

10:00..

- Opening / 連絡事項

10:15am..

Embedded Linux status update

- **Tim Bird** (Connect from USA)
  - Latest Community topics update - quick update.
  - Latest CE WG project updates
- (In English / 英語のセッションです)

- Media:Status-of-Embedded-Linux-2013-12-JJ47.pdf

11:30..

Lunch

12:30pm..

- **'Call for session'**

- Session proposal how-to. / 提案の方法

1:30pm..

- **'Public Viewing'**「カーネル コミュニティー - 参加者と活動内容　Greg Kroah-Hartman（The Linux Foundation、フェロー）」

- 横浜・赤れんが倉庫で開催中のイベントのセッションをUstream経由でパブリックビューイングします。

- http://events.linuxfoundation.jp/events/enterprise-users-meeting-japan/program/schedule
- http://www.ustream.tv/channel/eum2013 (Ustream)

2:00pm..

LTSI update

- Hisao Munakata

- http://elinux.org/images/5/59/Elce2013_LTSIansTESTING.pdf

2:30pm..

Introduction of Fujitsu's test cases (LTSI Workshop at Embedded Linux Conference Europe (ELCE) 2013)

- Teppei Asaba, Fujitsu

- http://ltsi.linuxfoundation.org/sites/ltsi/files/ltsi_workshop_20131025r4.pdf

3:15pm..

Expectation of LTSI Testing (LTSI Workshop at Embedded Linux Conference Europe (ELCE) 2013)

- Yoshitake Kobayashi, TOSHIBA

- Slides (in English)

- Slides (in Japanese)

4:00pm..

- **'Call for session'**

- Session proposal how-to. / 提案の方法

- Please be noted above time table is just a guideline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

- togetter

http://togetter.com/li/601944

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWikiに残してください。

English Translation Volunteer

- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

Categories:

- Japan Technical Jamboree
- Events

From: eLinux.org

Japan Technical Jamboree 48

The Linux Foundation CE Workgroup
Japan Technical Jamboree



Date: May 23rd/ 日付: 5月23日（金）

- At Nakano Sunplaza / 於、中野サンプラザ
- Please join "celinux-dev" mailinglist to get the latest announcement.
- "celinux-dev" メーリングリストにこのイベント関係のアナウンスが流れます。ぜひ参加してください。
  - How to join the mailing list.

Contents

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Workgroup of the Linux Foundation. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
  - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly without any complicated expressions. Most Japanese developers are capable to understand plain English.

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。
  - Japan Technical Jamboree 47

Hash tag on Twitter

- **#CELFJP**

Date and venue... / 日付・場所...

	• Date **May 23rd, 2014** ◦ **Starting at 10 am** • At ***Nakano Sunplaza / 8F Training Room*** / 会場 中野サンプラザ **/ 8**階・研修室 ◦ Nakano Sunplaza is located just close to **Nakano** station (JR/Tokyo Metoro). ◦ http://www.sunplaza.jp/ (Japanese) • Admission: ***Free of charge*** / 参加費用: 無料 • Not limited for CE Workgroup members. / CE Workgroup会員以外も参加・セッション持ち込み共に可能 • Coordinators / 世話役 (Your inquiries in English welcome) ◦ Hisao Munakata / 宗像尚郎 (hisao_dot_munakata_dot_vt(a)renesas_dot_com) ◦ Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

Registration / 参加登録

- No registration required / 参加登録は要りません。

Main Topics

- It is just after the Kernel Summit and final Jamboree before ELC Europe.

Agenda / 進行

Special Remarks

- 今回はいつもアメリカからSkypeで参加しているTim Birdさんは直接参加します。
- さらにLTSIのメンテナーでありDevice DriverのメンテナーでもあるGreg K-Hさんも参加してくれる事になりました！

Agenda

Time

Title and presenter

Notes

Presentation Materials

10:00am..

- Opening / 連絡事項

10:15am..

How The Linux Kernel is Developed / LTSI Update (tentative)

- Greg K-H, Hisao Munakata
- Video
- Video
- Greg's slide
- LTSI Update

11:30..

Lunch

12:30pm..

The Paradox of Embedded and Open Source

- **Tim Bird**

This is a repeat of Tim's keynote presentation from ELC. Can Linux be specialized for deeply embedded projects, as characterized by the Internet of Things, while still maintaining the network effects of community cooperation and sharing? Is this possible or even desirable?

- (In English / 英語のセッションです)
- Media:The-paradox-of-open-source-and-embedded-elc-2014.pdf
  - INCEPTION
- In Prezi format
- Video

1:30pm..

Brief Introduction of CEWGhttp://www.elinux.org/File:

- Video

1:45pm..

IPv6 Evaluation Report of LTSI-3.10

- Teppei Asaba, Fujitsu

http://ltsi.linuxfoundation.org/sites/ltsi/files/ltsi_workshop_20140430.pdf

- Video

2:00pm..

ARM 64bit has come!

- Tetsuyuki Kobayashi

Reports the first impression of A64 instruction set and execute them using QEMU.

- slide
- Video

3:45pm..

Android and 64 bit architecture (tentative)

- Hidenori Yamaji, Sony Mobile
- slide
- Video

4:45pm..

Using RT Preempt patch with LTSI-3.x

- TBD, TOSHIBA
- Slide1
- Slide2
- Video

5:45pm..

Evaluation of Real-time Property in Embedded Linux

- Hiraku Toyooka, Hitachi
- Slide
- Video

6:45pm..

Review ELC 2014

- Video

7:30pm..

- Please be noted above time table is just a guideline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

- togetter

http://togetter.com/li/670814 （Japan Jamboree #48)

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWikiに残してください。

English Translation Volunteer

- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

Categories:

- Japan Technical Jamboree
- Events

From: eLinux.org

Japan Technical Jamboree 49

The Linux Foundation CE Workgroup

Japan Technical Jamboree



Date: July 25th/ 日付: 7月25日（金）

- At Nakano Sunplaza / 於、中野サンプラザ
- Please join "celinux-dev" mailinglist to get the latest announcement.
- "celinux-dev" メーリングリストにこのイベント関係のアナウンスが流れます。ぜひ参加してください。
  - How to join the mailing list.

 We are calling for session proposal now! / セッション提案募集中！

Session proposal how-to. / 提案の方法

Contents

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Workgroup of the Linux Foundation. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
  - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly without any complicated expressions. Most Japanese developers are capable to understand plain English.

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。
  - Japan Technical Jamboree 48

Hash tag on Twitter

- **#CELFJP**

Date and venue... / 日付・場所...

<table>
<tr>
<td></td>
<td>

Date July 25th, 2014
Starting at 10 am
At Nakano Sunplaza / 7F Training Room 8 / 会場 中野サンプラザ / 7階・研修室8
Nakano Sunplaza is located just close to Nakano station (JR/Tokyo Metoro).
http://www.sunplaza.jp/ (Japanese)
Admission: Free of charge / 参加費用: 無料
Not limited for CE Workgroup members. / CE Workgroup会員以外も参加・セッション持ち込み共に可能
Coordinators / 世話役 (Your inquiries in English welcome)
Hisao Munakata / 宗像尚郎 (hisao_dot_munakata_dot_vt(a)renesas_dot_com)
Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

</td>
</tr>
</table>

Registration / 参加登録

- No registration required / 参加登録は要りません。

Main Topics

- It is just after the Kernel Summit and final Jamboree before ELC Europe.

Agenda / 進行

Agenda

Time

Title and presenter

Notes

Presentation Materials

10:00..

- Opening / 連絡事項

10:15am..

Embedded Linux status update

- **Tim Bird** (Connect from USA)
    - Latest Community topics update - quick update.
    - Latest CE WG project updates
- (In English / 英語のセッションです)

- Media:Status-of-Embedded-Linux-2014-07-JJ49.pdf
- Video

11:30..

Lunch

12:30pm..

- **'A New Era - Open Innovation in the Automotive Ecosystem with Linux and Open Source Community -'** (Repeat of a key note session of Automotive Linux Summit)
- **Masashige Mizuyama**, CTO of Automotive Infotainment Systems, Panasonic

- http://events.linuxfoundation.jp/events/automotive-linux-summit-spring/program/schedule

- http://monoist.atmarkit.co.jp/mn/articles/1407/04/news034.html (Summary, Japanese)
- Media:20140702_ALS_Keynote_Panasonic_Mizuyama.pdf

1:30pm..

- **'LTSI Project Update'**
- **Hisao Munakata**

- Video

2:30pm..

- **Linux File System Analysis for IVI System**
- **Mitsuharu Ito, Fujitsu**

- From Automotive Linux Summit

- http://events.linuxfoundation.org/sites/events/files/slides/linux_file_system_analysis_for_IVI_systems.pdf
- Video

3:30pm..

- **Google I/O 2014 Report**
- Katsuya Matsubara

4:15pm..

- **'Call for session'**

- Session proposal how-to. / 提案の方法

- Please be noted above time table is just a guideline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

- togetter

  - http://togetter.com/li/697556

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWikiに残してください。

English Translation Volunteer

- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

Categories:

- Japan Technical Jamboree
- Events

From: eLinux.org

Japan Technical Jamboree 50



Date: October 24th/ 日付: 10月24日（金）

- At Nakano Sunplaza / 於、中野サンプラザ
- Please join "celinux-dev" mailinglist to get the latest announcement.
- "celinux-dev" メーリングリストにこのイベント関係のアナウンスが流れます。ぜひ参加してください。
  - How to join the mailing list.

 We are calling for session proposal now! / セッション提案募集中！

Session proposal how-to. / 提案の方法

Contents

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Workgroup of the Linux Foundation. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
  - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly

without any complicated expressions. Most Japanese developers are capable to understand plain English.

50th!

- It is 50th!!

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。
  - Japan Technical Jamboree 49
  - http://togetter.com/li/697556

Hash tag on Twitter

- **#CELFJP**

Date and venue... / 日付・場所...

	• Date **October 24th, 2014** ◦ **Starting at 10 am** • At ***Nakano Sunplaza / 8F Training Room 6*** / 会場 中野サンプラザ ***/ 8階・研修室6*** ◦ Nakano Sunplaza is located just close to **Nakano** station (JR/Tokyo Metoro). ◦ http://www.sunplaza.jp/ (Japanese) • Admission: ***Free of charge*** / 参加費用: 無料 • Not limited for CE Workgroup members. / CE Workgroup会員以外も参加・セッション持ち込み共に可能 • Coordinators / 世話役 (Your inquiries in English welcome) ◦ Hisao Munakata / 宗像尚郎 (hisao_dot_munakata_dot_vt(a)renesas_dot_com) ◦ Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

Registration / 参加登録

- No registration required / 参加登録は要りません。

Main Topics

- It is just after the ELC Europe 2014 in Dusseldorf.

Agenda / 進行

Agenda

Time

Title and presenter

Notes

Presentation Materials

10:00..

- Opening / 連絡事項

- Media:SelfIntro50.pdf

10:15am..

- **Overcoming Obstacles to Mainlining**
- **Status of Embedded Linux**
- **Tim Bird** (Connect from USA)
    - Obstacles to mainlining and ways to overcome them (for large companies)
    - Latest Community topics update - quick update.
    - Latest CE WG project updates
- (In English / 英語のセッションです)

- Media:Overcoming_Obstacles_to_Mainlining-ELCE-2014-with-notes.pdf
- Media:Status-of-Embedded-Linux-2014-10-JJ50-v2.pdf
- Media:Android_upstreaming_status.pdf
- Video
- Video

11:30..

Lunch

12:30pm..

Short Report of ELCE/LinuxConEurope in Dusseldorf

- Shinsuke Kato, Panasonic Corporation

14時～17時，別件で退席します．なので午後一で枠を頂けると助かります． 内容的にも午後一で，気軽に聞いていただきたいものです．(時間は20～30分程度)

- (pdf/pptx 両方置いています. 15MBくらいあるのでご注意下さい)
- Media:ELCE2014_Report_Jamboree50_r6.pdf
- Media:ELCE2014_Report_Jamboree50_r6.pptx

13:00pm..

Introduction to intel DPDK

- Tetsuya Mukawa, IGEL Co.,Ltd.

- Media:Introduction_to_Intel_DPDK_v2.pdf

- Video

14:00pm..

IPv6 Evaluation Status of LTSI

- Yuki Machida, FUJITSU COMPUTER TECHNOLOGIES LIMITED

- Media:ipv6_evaluation_status_of_ltsi_r004.pdf

- Video

14:30pm..

Functional Safety

- Tadao Tanikawa, Panasonic
- Media:20141015_ELCE_FuncSafety_Workshop_public.pdf
- Video

15:30pm..

Introduction of slides, LTSI workshop at ELC Europe 2014

- Media:LTSIwsAtELCE2014.pdf

15:30pm..

- **Call for session**
- Session proposal how-to. / 提案の方法
- Please be noted above time table is just a guideline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

- togetter
  - http://togetter.com/li/736173

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWikiに残してください。

English Translation Volunteer

- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

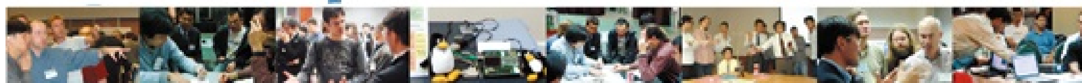
Categories:

- Japan Technical Jamboree
- Events

From: eLinux.org

Japan Technical Jamboree 51

The Linux Foundation CE Workgroup
Japan Technical Jamboree



Date: December 19th/ 日付: 12月19日（金）
- At Nakano Sunplaza / 於、中野サンプラザ
- Please join "celinux-dev" mailinglist to get the latest announcement.
- "celinux-dev" メーリングリストにこのイベント関係のアナウンスが流れます。ぜひ参加してください。
  - How to join the mailing list.

 We are calling for session proposal now! / セッション提案募集中！

Session proposal how-to. / 提案の方法

Contents

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Workgroup of the Linux Foundation. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
  - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly without any complicated expressions. Most Japanese developers are capable to understand plain English.

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。
  - Japan Technical Jamboree 50
  - http://togetter.com/li/736173

Hash tag on Twitter

- **#CELFJP**

Date and venue... / 日付・場所...

	• Date **December 19th, 2014** ◦ **Starting at 10 am** • At ***Nakano Sunplaza / 8F Training Room 6*** / 会場 中野サンプラザ ***/ 8***階・研修室***6*** ◦ Nakano Sunplaza is located just close to **Nakano** station (JR/Tokyo Metoro). ◦ http://www.sunplaza.jp/ (Japanese) • Admission: ***Free of charge*** / 参加費用: 無料 • Not limited for CE Workgroup members. / CE Workgroup会員以外も参加・セッション持ち込み共に可能 • Coordinators / 世話役 (Your inquiries in English welcome) ◦ Hisao Munakata / 宗像尚郎 (hisao_dot_munakata_dot_vt(a)renesas_dot_com) ◦ Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

Registration / 参加登録

- No registration required / 参加登録は要りません。

Main Topics

- Final Jamboree of year 2014.

Agenda / 進行

Agenda

Time

Title and presenter

Notes

Presentation Materials

10:00..

- Opening / 連絡事項

- Media:SelfIntro50.pdf

10:15am..

Overcoming Obstacles to Mainlining Status of Embedded Linux

- **Tim Bird** (Connect from USA)
    - Latest Status of Embedded Linux technologies
    - Latest CE WG project updates
- (In English / 英語のセッションです)

- Media:Status-of-Embedded-Linux-2014-12-JJ51.pdf
- Video

11:30..

Lunch

13:00pm..

- **LTSI Update**
    - Hisao Munakata
- Video

14:00pm..

- **Practice LTSI Test Framework & Introduction of ethtool Test Set**
    - Fan Xin, Fujitsu
- Media:LTSI Test Framework_r005.pdf

- Video

15:00pm..

- **Linux Filesystem Analysis for IVI Systems**
    - Teppei Asaba, Fujitsu
- Media:20141021_Rcar_filesystems.ver1.5.pdf

- Video

16:00pm..

- **Call for session**

- Session proposal how-to. / 提案の方法

- Please be noted above time table is just a guideline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

- togetter

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWikiに残してください。

English Translation Volunteer

- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

Categories:

- Japan Technical Jamboree
- Events

> From: eLinux.org

Japan Technical Jamboree 52

The Linux Foundation CE Workgroup
Japan Technical Jamboree



Date: April 10th/ 日付: 4月10日（金）

- At Nakano Sunplaza / 於、中野サンプラザ
- Please join "celinux-dev" mailinglist to get the latest announcement.
- "celinux-dev" メーリングリストにこのイベント関係のアナウンスが流れます。ぜひ参加してください。
  - How to join the mailing list.

 We are calling for session proposal now! / セッション提案募集中！

Session proposal how-to. / 提案の方法

Contents

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Workgroup of the Linux Foundation. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
  - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly without any complicated expressions. Most Japanese developers are capable to understand plain English.

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。
    - Japan Technical Jamboree 51

Hash tag on Twitter

- **#CELFJP**

Date and venue... / 日付・場所...

<table>
<tr>
<td></td>
<td>

- Date **April 10th, 2015**
    - **Starting at 10 am**
- At ***Nakano Sunplaza / 8F Training Room 6*** / 会場 中野サンプラザ ***/ 8***階・研修室***6***
    - Nakano Sunplaza is located just close to **Nakano** station (JR/Tokyo Metoro).
    - http://www.sunplaza.jp/ (Japanese)
- Admission: ***Free of charge*** / 参加費用: 無料
- Not limited for CE Workgroup members. / CE Workgroup会員以外も参加・セッション持ち込み共に可能
- Coordinators / 世話役 (Your inquiries in English welcome)
    - Hisao Munakata / 宗像尚郎 (hisao_dot_munakata_dot_vt(a)renesas_dot_com)
    - Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

</td>
</tr>
</table>

Registration / 参加登録

- No registration required / 参加登録は要りません。

Main Topics

- Just after the Embedded Linux Conference 2015 .

Agenda / 進行

Agenda

Time

Title and presenter

Notes

Presentation Materials

10:00..

- Opening / 連絡事項

- Media:SelfIntro50.pdf

10:15am..

Status of Embedded Linux

- **Tim Bird** (Connect from USA)
  - Latest Status of Embedded Linux technologies
  - Latest CE WG project updates
- Media:Status-of-Embedded-Linux-2015-04-JJ52.pdf (In English / 英語のセッションです)
- Video

11:30..

Lunch

1:00pm..

- **Poky meets Debian: Understanding How to Make an Embedded Linux by Using an Existing Distribution's Source Code**
  - Someone talked about this session
- File:Poky meets Debian Understanding How to Make an Embedded Linux by Using an Existing Distribution's Source Code.pdf
- Video

2:00pm..

- **Reviewing Embedded Linux Conference**
  - Shinsuke Kato (Panasonic)
  - (ELC/ABSのショートレポートを予定しています。時間は20〜30分くらいです。))
- Media:ELC_2015_Report_Jamboree.pdf
- Video

2:30pm..

- **Tricky implementation of Go ARM soft float**
  - Tetsu Kobayashi
- Video

3:00pm..

- **LTSI Update**
  - Hisao Munakata

4:00pm..

- **The Project: An Open Platform**
  - Rob Landley and his partners (The Open Processor Foundation)
- File:Jamboree R6.pdf
- Most of the presentation performed in English.
- Video

5:00pm..

- **Call for session**
- Session proposal how-to. / 提案の方法

- Please be noted above time table is just a guideline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

- togetter

http://togetter.com/li/806509

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWikiに残してください。

English Translation Volunteer

- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

Categories:

- Japan Technical Jamboree
- Events

From: eLinux.org

Japan Technical Jamboree 53

The Linux Foundation CE Workgroup
Japan Technical Jamboree



Date: June 19th/ 日付: 6月19日（金）

- At Nakano Sunplaza / 於、中野サンプラザ
- Please join "celinux-dev" mailinglist to get the latest announcement.
- "celinux-dev" メーリングリストにこのイベント関係のアナウンスが流れます。ぜひ参加してください。
  - How to join the mailing list.

 We are calling for session proposal now! / セッション提案募集中！

Session proposal how-to. / 提案の方法

Contents

Introduction / はじめに

- The Japan Technical Jamboree is a forum-wide technical meeting of the CE Workgroup of the Linux Foundation. This meeting will be located conveniently in Japan and use Japanese as the native language of the event.
- A general guidance page is available. / 初めての方はこちらもお読みください。
  - Japan Technical Jamboree Guidance (Japanese/English)

Special remarks for non Japanese speakers

- This page is Japanese/English bilingual. Please allow some contents on this page are not translated into English because of this event is Japan regional one, though we try to place English translation.
- If you would like to perform your presentation in English, we also welcome you to join! We hope you to speak slowly without any complicated expressions. Most Japanese developers are capable to understand plain English.

Previous Jamboree

- Please look into the wiki page. / 下記のWikiページをご覧ください。
  - Japan Technical Jamboree 52

Hash tag on Twitter

- **#CELFJP**

Date and venue... / 日付・場所...

<table>
<tr>
<td></td>
<td>

- Date **June 19th, 2015**
  - **Starting at 10 am**
- At ***Nakano Sunplaza / 8F Training Room 6*** / 会場 中野サンプラザ ***/ 8***階・研修室***6***
  - Nakano Sunplaza is located just close to **Nakano** station (JR/Tokyo Metoro).
  - http://www.sunplaza.jp/ (Japanese)
- Admission: ***Free of charge*** / 参加費用: 無料
- Not limited for CE Workgroup members. / CE Workgroup会員以外も参加・セッション持ち込み共に可能
- Coordinators / 世話役 (Your inquiries in English welcome)
  - Hisao Munakata / 宗像尚郎 (hisao_dot_munakata_dot_vt(a)renesas_dot_com)
  - Satoru Ueda / 上田理 (Satoru_dot_Ueda(a)jp_dot_sony_dot_com)

</td>
</tr>
</table>

Registration / 参加登録

- No registration required / 参加登録は要りません。

Main Topics

- Just after the Embedded Linux Conference 2015 .

Agenda / 進行

Agenda

Time

Title and presenter

Notes

Presentation Materials

10:00..

- Opening / 連絡事項

- Media:SelfIntro50.pdf

10:15am..

Status of Embedded Linux

- **Tim Bird** (Connect from USA)
    - Latest Status of Embedded Linux technologies
    - Latest CE WG project updates
- "PDF" (In English / 英語のセッションです)

11:30..

Lunch

1:00pm..

- **SPDX with Yocto Project**
- **Sharing & Reducing Test Efforts on AGL Kernels Using LTSI and LTSI Test Framework**
    - Fan Xin, Fujitsu
    - ALSのアンコールです。時間はどこでも構いませんので、勝手に動かしてください。
    - http://alsjapan2015.sched.org/event/961c768979b905034e465ce09e16a95e?iframe=no&w=i:100;&sidebar=yes&bg=no#.VXkyeby1nyw
    - http://alsjapan2015.sched.org/event/116cb34e86c529b89c011650b22b1dae?iframe=no&w=i:100;&sidebar=yes&bg=no#.VXkyYry1nyw
- "PDF"
- http://events.linuxfoundation.org/sites/events/files/slides/agl_2015_r006.pdf

2:00pm..

- **Reviewing Automotive Linux Summit / LinuxCon Japan**
    - Shinsuke Kato (Panasonic)
    - ALS/LCJのショートレポートを予定しています。時間は30分くらいです。
- Media:ALS_LinuxCon_Japan_2015_Report_Jamboree.pdf
- Media:ALS_LinuxCon_Japan_2015_Report_Jamboree.pptx

2:30pm..

- **Brief Report of LinuxCon / CloudOpen Japan 2015 (LT)**
    - Katsuya Matsubara, Tetsuya Mukawa (IGEL)
    - Lightinig Talkとして、LinuxCon / CloudOpen Japanの参加報告をします。
- No slide / 資料はありません。

3:00pm..

- **Meta-Debian: Extending Yocto Project's Poky**
    - Kazuhiro Hayashi, Toshiba
    - LinuxCon Japanの発表を日本語でします。
- "PDF"
- "Meta-debian handout"

4:00pm..

- **Call for session**
- Session proposal how-to. / 提案の方法
- Please be noted above time table is just a guideline and may be shifted. / 上記の時間割は目安です。かなり前後する可能性がありますので、あらかじめご承知おきください。

Special Remarks

- Please place some detail descriptions about each presentation topics.

こちらに各プレゼンテーションの詳細などを記載してください。

- togetter

http://togetter.com/li/806509 (前回分)

Ask for your help / お願い

Presentation Materials

- We wish you to prepare the materials in English. / 出来るだけプレゼンテーション資料は英語で表記してください。絶対ではありませんが、日本語が理解できない方に対しての配慮が出来ればと思います。
- Please leave your material in this wiki site after the event. / ジャンボリー終了後、プレゼンテーション資料はこのWikiに残してください。

English Translation Volunteer

- If you can help the translation volunteer from Japanese to English, we would be very much appreciated! / 日本語を英訳していただくボランティアを大歓迎します！

Categories:

- Japan Technical Jamboree
- Events

> From: eLinux.org

Kernel Summit 2009

2009

The 2009 Kernel Summit was held in Tokyo, Japan, October 18-20

There was an end-user panel which included 3 embedded presentations.

You can find the presentation from Sony here:

- Media:Kernel-summit-2009-end-user-panel-Sony.pdf

Categories:

- 2009
- Events
- KS

> From: eLinux.org

Long Term Support Kernel Meeting 2011

This is information about the Long Term Support kernel meeting planned for Tuesday, October 25th in Prague, Czech Republic.

It is co-located with Embedded Linux Conference Europe in the Clarion Congress Hotel.

Contents

Purpose

This meeting will be to share latest status of LTSI, schedule and so on with getting together with CE industry's related party. Hosted by CEWG/Linux Foundation.

Details

- Name of the meeting: LTSI internal meeting or maybe kickoff meeting
- Date/Time: October 25, 1:30-3:00?
- Place: *(to be confirmed)* Room, Clarion Congress Hotel, Prague, Czech Republic

Invitees

CEWG will send invitation letter to key member of this project.

Agenda

- Introduction of LTSI
- Q&A
- Message from the Linux Foundation
- Message from partners, maybe Qualcomm, Linaro, Greg
- Announce further schedule for next step

Action Items

```
0. Review this proposal and add any items.
1. We will send invitation letter sooner. They may need to change travel schedule.
2. Negotiate each companies and others before the meeting
3. Ask some of key partners to give us message at the meeting
4. Discuss with Linaro for sharing QA activities
```

Attendees

Here is a table of confirmed attendees for the meeting:

Name	Company	Notes
Tsugikazu Shibata	NEC	Leader of the LTSI project
Hisao Munakata	Renesas	Leader of the LTSI project, AG co-chair
Tim Bird	Sony Network Entertainment	CEWG AG Chair
Satoru Ueda	Sony	

Possible attendees

- Hiroshi NOZUE - Toshiba
- Yoshitake KOBAYASHI - Toshiba

Categories:

- LTSI
- 2011
- Events
- Presentations

> From: eLinux.org

LTSI workshop in Osaka

日付: 6月15日（金）16時～

ATC Hall B5会議室 (コスモスクエア経由 トレードセンター前駅 下車すぐ)

Introduction / はじめに

- LTSI のワークショップを大阪で開催します！ LTSIについて、その詳細を知る絶好の機会です！
- LTSI については こちらを参照ください・LTSI home

概要

- 日時 **6**月**15**日 金曜日**, 2012**
  - 午後**4**時 スタート
- 場所 ***ATC Hall B5***会議室
  - ATC Hall (コスモスクエア経由 トレードセンター駅前) http://www.atc-co.com/atchall/access/index.html
  - B5会議室(会議室の場所 http://www.atc-co.com/atchall/conference/index.html)
- 参加費用: 無料
- どなたでも参加可能です
- 参加登録は要りません。
- 参考情報
  - 4月17日のLTSI Workshop @Yokohama の模様 (https://jp.linux.com/linux-community/eventsreport/388744-ltsi-partner-mmeting)

- 当日の資料です
  - https://events.linuxfoundation.org/images/stories/pdf/lcjp2012_munakata.pdf

Category:

- Events

OLS2004

Contents

Ottawa Linux Symposium 2004

http://LinuxSymposium.org/2004/

News

Audio for the keynote is online. See below.

Other Links

- http://www.linuxsymposium.org/2004/schedule_static.html
  - schedule
- http://www.finux.org/proceedings/
  - proceedings
- http://www.finux.org/Reprints/ - reprints
- http://www.finux.org/audio/ - keynote and other stuff as soon as we get it there.

Gallery Links

- http://rikers.org/gallery/ols2004
- http://gate.crashing.org/~benh/pics/OLS2004 (updated jul. 25)
- http://www.jebus.ca/gallery/OLS_2004
- http://www.heinous.org/images/2004-07_OLS/
- Hacker bike ride: http://vic.dyndns.org/pics/2004-07-25/index-simple.html
- http://ols2004.qbang.org - contains both fatal's and taj's pictures now (changed aug. 6th)
- taj suite 1903 (), camping http://www.qbang.org/ols2004/ (mirrored above)
- http://photos.jonmasters.org/OLS_2004
- http://www.ludusdesign.com/gallery/ols2004
- http://www.yak.net/random/pics/2004.07.21.OLS2004/
- http://people.nit.ca/~dcoombs/ols2004/
- http://gnumonks.org/static/photos/ks2004/
  - kernel summit group picture
- http://flickr.com/photos/sfllaw/sets/202883/
- http://gallery.jukie.net/2004-OLS
- http://photos.pavlov.net/gallery/168542
- http://fooishbar.org/gallery/ottawa-ols-jul04
- http://gallery.yeoh.info/gallery/album33
- http://zeevon.dyndns.org/gallery/ols2004

Please add more links.

Category:

- Events

> From: eLinux.org

OLS 2007 CELF BOF

This BOF was held on Friday, June 29, at the Westin Hotel in Ottawa, Canada (at the Ottawa Linux Symposium).

Agenda

Person	Topic
Matt Mackall	Discussion and demo of pagemap (accurate memory measurement) patches and tools
Michael Opdenacker	http://www.elinux.org/images//8/83/Pdf.gif Size and Bootup Time techniques and demo http://www.elinux.org/images/d/da/Info_circle.png, Media:Ols2007-qemu-arm-demo.odp (OpenOffice.org format)
Tim Bird	http://www.elinux.org/images//8/83/Pdf.gif Realtime Testing for Embedded Platforms http://www.elinux.org/images/d/da/Info_circle.png- Discussion on recent RT-preempt testing, with techniques and issues.
Manas Saksena	Fedora on ARM project. To get involved, subscribe to the fedora-arm mailing list.
Satoru Ueda	Lightning talk on Japan Jamborees, and CELF white paper to management
Buffalo	Lightning talk on KuroBox/Pro
Algo	Lightning talk on Algo Smart Display
Rob Landley	Lightning talk on native compiling vs. cross compiling
Klaas van Gend	Lightning talk on Embedded Linux Conference, Europe
Steve Rostedt	Lightning talk on realtime testing
???	Lightning talk on Mamona build system

Report

[*Anyone care to do a writeup?*]

Major prizes - ARM-based development boards - were won by **Adam Belay** and **Ruud Derwig**.

Categories:

- Events
- 2007

OLS 2007 Embedded Linux BOF

Description

Date: June 27, 6-7 pm, King room

Leader: Tim Bird

This BOF will review the overall state of embedded Linux. This will include describing changes in the Linux kernel in the last year related to embedded processors and products, as well as discussions of notable embedded products using Linux in the last year. As usual, we will commiserate on the sorry state of affairs, celebrate our progress, and plan for how to wrest control of linux from those pesky desktop, server and enterprise developers.

Presentation

http://www.elinux.org/images//8/83/Pdf.gif Embedded Linux BOF presentation
http://www.elinux.org/images/d/da/Info_circle.png

Categories:

- Events
- 2007

> From: eLinux.org

OLS 2007 Embedded Linux Wiki BOF

Contents

Description

Date: June 27, 7-8 pm, King room

Leader: Michael Opdenacker

The new Embedded Linux Wiki (elinux.org) aspires to become the leading resource of information about Embedded Linux. The project has been seeded with the large quantity of technical material currently maintained on the CELF public wiki and current elinux.org wiki. The CE Linux Forum (CELF) is a main sponsor, funding a fulltime editor for the site. The wiki will also welcome and encourage contributions beyond technical documentation, such as hacker, product and company profiles, event coverage, interviews, book reviews...

This BOF reviewed the achievements of this project to date and provided an overview of the ideas and plans which have not been implemented yet. All participants were invited to discover the work to date and share their expectations of what a useful online resource should provide.

During the BOF, Michael asked everyone in the audience to create an account and "edit something, anything". This is an editing contribution from one of those audience members.

BOF report

Presentation slides: http://elinux.org/images//8/83/Pdf.gif Embedded Linux Wiki BOF presentation http://elinux.org/images/d/da/Info_circle.png, Media:Ols2007-elinux-wiki-bof.odp (OpenOffice.org format)

During the presentation, several questions we asked:

Q: Where are the IRC archives? A (Tim Riker): http://ibot.rikers.org/%23elinux/ (now listed on the IRC page)

Q: What about email addresses? A: Registration doesn't ask for any password.

Q: What's the license for code excerpts? A: The license of the original software. If the code doesn't belong to any project, the author is free to choose a license. If not specified, we assume the license is the GNU GPL v2.

Q: Are links stable over time? A (Tim Bird): Yes, if the page is old enough. After a new page is created, you may expect us to rename it or merge it with another page if appropriate.

Q: Could there be an RSS feed for pages? A: Apparently not, for new pages. However, Wikipedia provides an RSS link in the "Toolbox" box in each history page. We are using the same engine (Wikimedia), but we have not enabled this feature, yet.

Q: What about Wiki promotion activities and prizes? A (Tim Bird): They are not finalized yet. Stay tuned on the Wiki pages for news.

Several suggestions were also made:

- Page(s) in which user could share their live experience with systems that appear to run Linux. An example is the entertainment system used by Delta Air Lines. There could be one page per system, in which contributors could add the extra details they found.
- Rob Landley: as a way to fight obsolete pages, old pages could carry a "not modified since..." notice. Of course, contents would not be removed.
- Rob Landley: To attract and recognize contributors, some pages could be rewarded by a "Passed editorial approval" or "High quality article" notice.
- Rob Landley: What about attracting people by trying to catch their attention every day (note: like in Wikipedia)? We could have weekly highlights to keep contributors active. Tim Bird: I rather see this wiki as a live Embedded-Linux book.
- Have "Questions and Answers" pages on several topics, perhaps on each topic advertized on the front page. Rob Landley also suggested having an "Unanswered questions" section on these pages or on separate pages.
- Have tool-specific documents, such as "Installing SNMP".
- Somebody (Lefty, IIRC) proposed to announce the new wiki on Slashdot, LWN.net, and LinuxDevices.com.
- Somebody proposed holding meetings with contributors living in the SF Bay area.
- Satoru Ueda told us that there could be more contributors in Japan, but they could only write in Japanese. He suggested having a section or version of the wiki in Japanese for these contributors. Rob Landley suggested having a "Japanese Maintainer" who would coordinate and review translations of these pages to English.

Additional questions, answers and suggestions

Even if you couldn't participate to this event, do not hesitate to add your own questions and suggestions below.

Multiple Languages

Just responding to the note about Japanese contributions above... is it not possible to set up mediawiki so that we can have a "in other languages" sidebar just like wikipedia does? I'm sure there will be some language specific pages, but I think most content could just be organized so there is a version of each page/topic for the various languages.

Categories:

- Events
- 2007

> From: eLinux.org

OLS 2008 CELF Embedded Developer BOF

Here is information about the Embedded Developer BOF (Birds-Of-a-Feather) meeting that CELF hosted at the 2008 Ottawa Linux Symposium.



CELF hosted the meeting on Friday, July 25 at the Westin Hotel in Ottawa. As near as we could count, there were about 120 people in attendance.

Tim Bird, the CELF Architecture Group Chair led the meeting. There were 4 mini-sessions and a few lightning talks, followed by some prize giveaways.

Presentations

- Best Of Recent CELF Conferences presentation by Tim Bird
- Developing Embedded Linux With Target Control by Tim Bird
  - Target Control home page is here: http://mirror.celinuxforum.org/labwiki/Target_Control
- Embedded tools survey by Michael Opdenacker
- GCC Tips and Tricks Highlights by Gene Sally
  - See also GCC Tips





Prizes

- Tin Can Tools Nail Kit
- 5 Inaura Black Dog security devices
- Sony Mylo2 (donated by TimeSys)
- Sony PS3 (with screen and keyboard)
- HP Media Vault 5150 (1.5 TB NAS device)

Categories:

- Events
- 2008

> From: eLinux.org

Ottawa Linux Symposium 2006

Here is information about CE Linux Forum's involvement and participation in Ottawa Linux Symposium 2006.

Table Of Contents:

CELF had big involvement in OLS 2006. CELF was a sponsor of OLS and we had a number of meetings at and around the event. There were presentations and BOFs by many CELF members. Also, we continued our tradition of handing out prizes at the closing keynote address.

Here are some of notes and presentations from the event.

Contents

Embedded Linux BOF

-
  - Wednesday, July 19, 5:00 pm to 5:45 pm, Congress Centre, Room C

In this BOF, Tim gave his "State of Embedded Linux" talk. He described some what's happened with Linux in the last year, related to its use in embedded devices. This includes an overview of things recently mainlined (like Linux-tiny patches), as well as ongoing work and research in areas like power management for embedded devices, system memory size, bootup time improvements, realtime, measurement tools, etc.

Here is Tim's presentation: Media:OLS2006-Embedded-BOF-2.ppt

Here are **PORTIONS** of Tim's presentation, with links to relevant articles (mostly on lwn.net) /\ - please see the file for the full presentation.

-
  - Realtime
    -
      - hrtimers (2.6.18?)
      - clock sources (2.6.18)
      - HR timer API

- http://lwn.net/Articles/167315/
- ktimers explanation:
  - http://lwn.net/Articles/152436/
- Generic IRQ (2.6.18)
- Tickless Idle coming

- Rt-Preempt
  - Mainline status
    - Most stuff in, only need:
      - sleeping spinlocks
      - threaded IRQs
  - Priority Inheritance (2.6.18)
    - http://lwn.net/Articles/177838/
  - Latency tracer
- Unit-at-a-time Compilation
  - Ingo Molnar patch to utilize gcc 4 unit-at-a-time compilation
    - http://lkml.org/lkml/2005/12/28/68
    - http://lwn.net/Articles/165354/
- inline reduction
  - New _always_inline_ attribute:
    - http://lwn.net/Articles/167315/
    - http://lwn.net/Articles/165354/
- Size Testing
  - Bloatwatch - http://testlab.celinuxforum.org/bloatwatch/
  - Config size testing - Presentation at CELF Project BOF
- Security
  - App'***Armour***
    - Lighter than SELinux

            - Good enough for embedded work??
        - LSM in peril?
    - Bootup Time
        -
            - XIP - Execute in Place
                -
                    -
                        - Old info: http://lwn.net/Articles/135472/
            - XAFS - new file system specifically for XIP
                -
                    -
                        - Jared Hulbert (Intel) posted for comments: http://lwn.net/Articles/182337/
    - Power Management
        -
            - User-space software suspend
                -
                    - Latest kerfluffle over suspend phases
                    - Linus has posted a new patch, showing his method.
                    - See lwn.net: http://lwn.net/Articles/189467
            - Linux PM summit - http://lwn.net/Articles/181687
    - Audio/Video/Graphics
        -
            - Big direct rendering update - http://lwn.net/Articles/167315/
            - New CELF AVG Spec. (2.0)
                -
                    - DirectFB, ALSA, OpenGL/ES, UHAPI
    - Mobile phone stuff
        -
            - CELF specification still in-progress
            - Many orgs: CELF MPPWG, OSDL MLI, LiPS Forum, the new foundation
    - Tools
        -
            - Tracing - LTTng just recently re-added lots of architecture support
            - Memory Leak Tracker
            - Function re-ordering - http://lwn.net/Articles/173657/

Some of the major discussion items were:

-
    - standardization of GUI
        -
            - Why can't CELF standardize on a single embedded GUI? The answer is that the space is already fragmented, and getting product vendors to switch UIs will be very difficult. Different capabilities of the existing solutions in this area (Qt, GTK, and X), as well as different capabilities for related software (such as apps) led to this fragmentation.
            - There's not an easy answer, but maybe if the field tilts towards one platform, it will then begin to dominate.

- mobile phone forum fragmentation
    -
        - CELF is working with the LiPS Forum and OSDL/MLI, don't know much about the new organization. So far, organizations are cooperating and avoiding conflicts.
- size issue
    -
        - Tim took an informal survey of the audience, asking what was the number one problem using Linux: About 90% answered "size". Power management and bootup time were also considerations.
        - Many linux-tiny patches have been merged over the last year.
        - There's a new system called "bloatwatch", where kernel size is automatically tested each new release. Developers can examine the results to see the size of subsystems and symbols. Sizes can be compared between versions, and are graphed over time. See
- corporate developer participation in community
    -
        - Developers from corporations often don't participate effectively in the open source community Several issues contribute to this: 1) product treadmill, 2) getting stuck on old versions, 3) not enough time budgeted by company for this activity, and 4) product deadlines leading to (ahem) less-than-mainlineable code. There are no clear answers, but CELF will continue encouraging direct participation in open source by its members.
        - Tim took an informal (show of hands) survey, and very few respondents have yet to ship a 2.6-based product. Some companies are still working on 2.4 or previous kernels.

CELF Project BOF

-
    - Friday, July 21, 7:00 pm to 9:00 pm, Les Suites Hotel

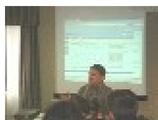
At this meeting we discussed some of CELF's initiatives and member company projects.

The following presentation introduced the topics, and had information about the CELF Test Lab: Media:CELF-Projects-BOF.ppt

Quickie CELF Initiatives Overview

-
    - Tim described some of the initiatives and projects that CELF is directly managing and funding:
        -
            - Linux-tiny mainlining, and bloatwatch (via Selenic Consulting - Matt Mackall)
            - UHAPI/DirectFB integration (via Dennis Kropp)
            - CELF Open Test Lab (via Nomad Global Solutions)
            - Embedded Linux Conference - to be held April 17,18,19, 2007 in San Jose, California, USA

CELF Test Lab Introduction and Demonstration

By Tim Bird, Sony Electronics 

-
    -
        -
            - Basic lab infrastructure is now set up, and tests can be run remotely on multiple boards.

- Tim demonstrated running a test in the lab using the web interface, as well as showed interactive use of the lab via ssh to a host and target in the lab.

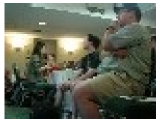
See more about the lab in the BOF slides (see above).

Config Weight Size Test

By Munehiro Ikeda, NEC 

- Ikeda described his examination to make clear the size impact (image size and memory effect) of different kernel config options.
  - Examination tool named "Kconfig Size"
  - Current examination result summary
  - New
  - Further works plan
- Presentation material : Media:size_exam_celf_2006-07-21.pdf

CABI (CPU Resource Management)

By Midori Sugaya, Waseda University 

- Sugaya described the requirements of embedded system especially about resource managements in Linux, then proposed CABI and showed its function, architecture and enhancements (CPU Reservation). The contents of the presentation are following.
  - Resource management requirements of embedded system
  - Limitations of Linux scheduler
  - Proposed system : CABI (CPU Accounting and Blocking Interfaces)
  - The accounting model and architecture
  - CPU Reservation (priority boost approach)
- Presentation material : Media:2006_0721_LinuxSympoisum_clinux_bof_CABI_pdf.pdf

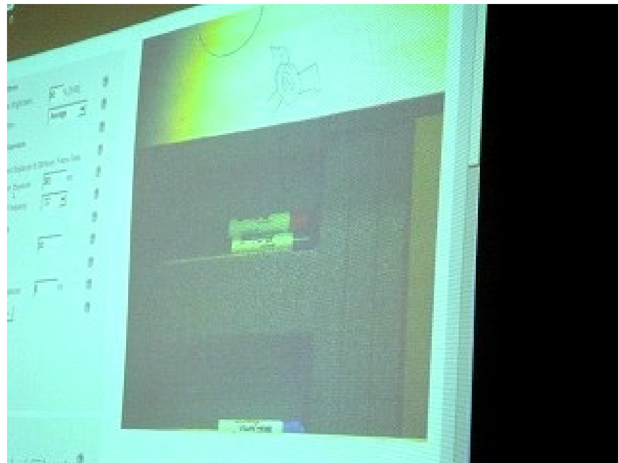
Annual CELF "Challenge" Item

Discussion led by Tim.

- We discussed CELF's proposal for creating a list of technology projects to focus on for the year
  - How can CELF push the projects forward?
    - Incentives - support, hardware, prizes, funding, labor?
- What projects deserve focused CELF attention??
- Does anyone need hardware that CELF or it's members can provide?

- Should we do a "hackfest"?

Hi-definition camera demonstration by Lumenera

 Demonstration scene.	 "Picture of bird by Tim Bird." Actual shoot of the camera.

[need to put info here]

Jamboree Report

By Satoru Ueda,

- Ueda-san talked about CELF's regional Jamborees being held in Asia (primarily Japan).
  - see Japan Technical Jamboree Guidance, Japan Technical Jamboree 8 and Japan Technical Jamboree 9
  - **The next Japan Jamboree**
    - August 25, in Tokyo
    - Detail is described in Japan Technical Jamboree 10

Presentation material:

- Media:JapanJam_OLS2006.pdf

Architecture Group meeting

- Tuesday, July 18, 3:00 pm to 8:00 pm at the Les Suites Hotel

This meeting was open to CELF Architecture Group members and invited guests.

FIX ME! WHERE ARE THESE LINKS: See Celf Wiki:Ag July 2006Meeting for the presentation and meeting notes for this meeting.

OLS Sessions by (or including) CELF members

Employees of CELF member companies made a number of presentations this year at OLS. Also, we led or participated in a number of BOFS. Here is a list of presentations and BOFS where CELF members presented or discussed topics important for embedded developers:

-
    - A Reliable and Portable Multimedia File System - Jaekyoung Bae, Samsung, Wed. July 19 12:00, Room B
    - Power Management BOF - Mark Gross, Intel, CELF PMWG Chair, Wed. July 19, 7:00 pm, Room A
    - Tracing BOF - William Cohen, RedHat, (Tim Bird, Sony will also present), Wed. July 19 7:00 pm, Room D
        -
            - Tim's presentation for this is here: Media:OLS2006-Tracing-BOF.ppt
    - The effects of filesystem fragmentation - Ard Biesheuvel, Philips, Thur. July 20, 10:00 am, Room A
    - Tutorial: Porting a 2.4.20 character driver for legacy hardware to 2.6.15 - Mark Gross, Intel, Thur. July 20, 10:00 am, Room A
    - Improving Linux Startup Time Using Software Resume (and other techniques) - Hiroki Kaminaga, Sony, Fri. July 21, 10:00 am, Room A
        -
            - Presentation material is here: Media:snapshot-boot-final.pdf
            - Demonstration Video: /\ *Large size files (AVI). MP4 files are iPod(TM) compatible.*
                - startup using Resume From Disk: Media:SWSUSP.AVI (14.5MB) / Media:SWSUSP.MP4 - to obtain these videos, please email the administrator of this site.
                - Proposed Snapshoot Boot: Media:SSBOOT.AVI]] (8.0MB) / Media:SSBOOT.MP4 - to obtain these videos, please email the administrator of this site.
    - Linux Bootup Time Reduction for Digital Still Camera - ChanJu Park, Samsung, Sat. July 22, 12:00 pm, Room A

The full conference schedule is available at: http://www.linuxsymposium.org/2006/schedule.php [Should change this to link to proceedings, when they are online]

/\ *Please note that several other sessions at OLS reflect work that is sponsored or* funded (in whole or in part) by CELF member companies.

Categorized Session List

CELF produced a guide to OLS sessions for embedded developers]. This might help you look through the OLS 2006 proceedings for presentations of interest. The Proceedings are available at: http://www.linuxsymposium.org/2006/proceedings.php

Demos

-
    - July 19, 10:45 am to 6:00 pm, Congress Centre, 1st floor

CELF members showed a number of demos of their current open source projects. Come see live demonstrations of technology developed for using and customizing Linux in Consumer Electronics products.

-
    - See OLS2006Demos for details.
    - ;)) Photos uploaded

Prizes

CELF had several prizes to hand out again, as a way of saying "Thank You" to the open source community.

-

  - Philips settop box development platform
  - Nokia 770 Internet Tablet

The prizes were handed out by Tim Bird at the final keynote:

-
  - Date: Saturday, July 22
  - Time: ~5:00 pm (Keynote starts at 4:00)
  - Place: Congress Centre, Room A

Categories:

- Events
- 2006

> From: eLinux.org

Ottawa Linux Symposium 2007

Here is a list of activities sponsored by or of interest to CELF members at OLS 2007:



CELF Embedded Linux Projects BOF.
Thanks for so many developers attending!

Contents

Overview

The CE Linux Forum is a sponsor of the 2007 Ottawa Linux Symposium.

This is our third year of sponsorship, and we really enjoy participating each year. Our participation consists of several items:

- CELF hosts its own BOF, which is open to the public
- Individual members present specific sessions at the event
- CELF members often lead or participate in BOFs
- CELF gives out prizes donated by member companies at the final keynote session of the symposium.

 *Note: The acronym "BOFs" stands for "Birds-of-a-Feather"session.*
This is an informal session where people with similar interests get together to share their ideas and plans.

CE Linux Forum Bird's of a Feather session

This BOF is for anyone interested in the latest work being done in the area of embedded Linux. We have a few interesting mini-talks on new memory measurement tools, size and bootup time reduction techniques, and experience with testing realtime features in Linux.

Please feel free to come and discuss your own embedded project.

The BOF was held at:

- Place: Westin Hotel, adjacent to the Congress Centre
- Room: Confederation Ballroom III, on the Fourth Floor
- Date: Friday, June 29
- Time: 7:00 pm to 9:00 pm

CELF BOF Agenda:

Person	Topic
Matt Mackall	Discussion and demo of pagemap (accurate memory measurement) patches and tools
Michael Opdenacker	Size and Bootup Time techniques and demo
Tim Bird	Discussion on recent RT-preempt testing on embedded platforms
Manas Saksena	Fedora on ARM project
<open floor>	Lightning talks, demos, etc. Come and tell us about your own embedded project!
	Lightning Talks from Japan: See CELF wiki page for OLS2007 for links to lightning talk papers and presentations

Sessions and BOFS on embedded topics

Embedded Linux BOF

- Led by Tim Bird, this is a look at the current status of embedded features in Linux.
- Date and Time: Wednesday, June 27 at 6:00 pm / Room: King
- See http://www.linuxsymposium.org/2007/view_abstract.php?content_key=5

Embedded Linux Wiki BOF

- Led by Michael Opdenacker, this is a discussion about the new embedded Linux wiki.
- Date and Time: Wednesday, June 27 at 7:00 pm / Room: King
- See http://www.linuxsymposium.org/2007/view_abstract.php?content_key=268

BoFS: Linux in Mobile Phones

- Led by Scott Preece, this BOF summarizes work on Linux in Mobile phones
- Date and Time: Friday, June 29 at 6:00 pm / Room: Rockhopper
- See http://www.linuxsymposium.org/2007/view_abstract.php?content_key=250

Power Management BoFS

- Led by Mark Gross, this BOF discusses the latest developments and plans for power management in Linux
- Date and Time: Thursday, June 28 at 6:00 pm / Room: Rockhopper
- See http://www.linuxsymposium.org/2007/view_abstract.php?content_key=26

Sessions by category

Realtime sessions

- Internals of the RT Patch, by Steven Rostedt of RedHat

Power management

- Power Management BOFs
  - Mark Gross
- ACPI in Linux -- Top 10 Myths vs. Reality
  - Len Brown

Security

- App'***Armour Application Security BOFs - Seth Arnold***
- TomoyoLinux BOF
  - ToshiharuHarada
  - TOMOYO Linux BoF - Ottawa Linux Symposium 2007
  - http://tomoyo.sourceforge.jp/wiki-e/
  - http://tomoyo.sourceforge.jp/wiki-e/?TOMOYO-LSM
  - http://tomoyo.sourceforge.jp/wiki-e/?TOMOYO-GUI
  - http://lkml.org/lkml/2007/6/13/58
  - http://tomoyo.sourceforge.jp/cgi-bin/lxr/source/
  - http://sourceforge.jp/projects/tomoyo/document/elc2007-presentation-20070418.pdf/en/4/elc2007-presentation-20070418.pdf (ELC2007 slides)
- Linux Integrity Projects
  - David Safford
- Trusted Secure Embedded Linux: From Hardware Root of Trust to Mandatory Access Control
  - Hadi Nahari

Tracing

- Linux Tracing BOFs
  - Vara Prasad
- Djprobe - Probing the Kernel With the Smallest Overhead
  - Masami Hiramatsu
- In-kernel Dynamic Application Tracing Mechanism and Dynamic Tracing of Kernel Data Structure
  - Prasanna S. Panchamukhi
- Dynamic Tracing and Performance Analysis Using SystemTap
  - Mike Mason
- Ptrace, Utrace, Uprobes: Lightweight, Dynamic Tracing of User Apps
  - James A. Keniston

Bootup technology

- Breaking the Chains: Using LinuxBIOS to Liberate Embedded X86 Processors
  - Jordan H. Crouse
- Readahead: Time Travel Techniques For Desktop and Embedded Systems

- Michael Opdenacker

System Size

- Evaluating Effects of Cache Memory Compression on Embedded Systems
  - Anderson Farias Briglia
- More Linux for Less
  - Robin Getz

Architecture/Processor Support

- Distributed Cluster Computing on Cell/PS3 slide:attachment:ols-2007-ps3-bof.pdf - Akira Tsukamoto
- Linux on Cell Broadband Engine
  - Arnd Bergmann

Tools

- Cross Compiling Linux
  - Rob Landley
- The 7 Dwarves: Debugging Information Beyond gdb
  - Arnaldo Carvalho de Melo
- Frysk 1, Kernel 0?
  - Andrew Cagney
- Where is Your Application Stuck
  - Vivek Kashyap
- Implementation of a Branch Tracing Tool and Regression Test Framework
  - Hiro Yoshioka

CELF Areas of emphasis

These are the areas of emphasis that CELF is promoting at this year's symposium:

- RT-preempt - what can we do to help improve it and get it mainlined?
- Embedded Linux Wiki -

Categories:

- Events
- 2007

> From: eLinux.org

Proposed OSCON 2012 Embedded Linux track

OSCON 2011 had a limited amount of content about Linux, embedded development, BSD and relevant tools, notably in the Open Hardware track. Let's propose a lot more talks and sessions in 2012. The purpose of this page is to gather suggestions for what embedded Linux content OSCON 2012 should include.

Contents

Motivation: Why the Embedded Community Might Care about OSCON

OSCON is a giant meeting of a diverse group of influential open source developers, leaders, and businesspeople. By speaking to contributors to other projects outside our traditional contacts in the embedded world, eLinux engineers can recruit talented people who are not currently working on embedded and can promote the spread of Linux into new applications. For example, OSCON attracts Python developers who might attend one presentation about build and configuration management in the embedded space but who never attend ELC. OSCON attendees who work on server-based virtualization might be curious to learn more about device emulators or mobile security. UI/UX designers in Portland might chat in the hallway with compiler deployers who are stuck with creating a new GUI . . .

Strawman Suggestions for Presentations

- Satellite meetings
- Dedicated tracks
- Invited talks
- Sessions
- Contributed talks
- Workshops

Satellite meetings

Dedicated tracks

Invited talks

- "High-Level Web Interface to Low-Level Linux: the Case of the Beagleboard", Jason Kridner, TI
- Java Virtual Machines: design considerations for mobile
- New Features in Qt5 and What They Offer for Mobile
- KDE or GNOME on Mobile Devices

Sessions

- Build systems (featuring embedded Linux)
- Mobile embedded and its challenges (security, battery life . . .)
- Automotive Linux and Intelligent Transportation Systems

Contributed talks

Workshops

- BeagleBoard
- Arduino
- Yocto

> From: eLinux.org

Technical Conference 2005

Table Of Contents:

Contents

Presentation Materials

The conference is over, but you can see most of the (excellent) presentations at: Tech Conference 2005Docs

Introduction

The CE Linux Forum is holding a technical conference (in conjunction with their annual Plenary meeting).

- Location: San Jose - Dolce Hayes Mansion (South San Jose area - see map in Hotel Information below)
- Date: January 25-26, 2005 (Tuesday-Wednesday)
- Times:
  - January 25: 2:00 pm to 7:00 pm half-day
  - January 26: 9:00 am to 6:00 pm full-day
- Snacks and Meals:
  - Drinks and snacks will be available throughout both days
  - Lunch and dinner will be provided on both days
  - Drinks, snacks and the meals listed are included in your registration (that is, they're free)

Registration

Members of the forum should register at: Plenary Registration.

If you are an employee of a member company then you can register as a CELF member here, and then register for the technical conference using the link above.

(See the CELF member company roster here.)

Special non-member invited guests (mostly, people Tim Bird has talked to) will be registered by the forum office.

While openings are available, member companies may invite non-member guests. Please limit the number of non-member guests to 2 per company (due to costs). Please limit the total number of attendees from your company (including non-member guests) to 8.

/\ In order for us to make appropriate arrangements for the conference facilities, please register as soon as possible. Space is filling up - so don't delay.

Sessions

There will be 28 sessions, and Tuesday evening there will be a over 20 technical demonstrations presented by various members of the forum.

The sessions will consists of:

- Presentations
- Discussions
- A Panel
- Tutorials

Demonstrations should be registered here (at the bottom of the page).

Session List

Technical sessions are as follows (in no particular order):

Person	Session Description	Session Slot
Greg Ungerer, SnapGear	uCLinux	25-2:00-A
Tohru Nojiri, Hitachi	Linux Kernel State Tracer (LKST) technology	25-2:00-C
Todd Poynor, Monta Vista	Dynamic Power Management	26-10:00-B
Philippe Robin, ARM	Developing and Optimizing Linux on ARM Cores	25-3:00-A
Ed Plowman	OpenGL ES, Open VG and Open MAX	26-9:00-C
Matt Mackall	Linux-tiny and Directions for Small Systems	26-11:00-A
Erik Andersen	uClibc	26-10:00-A
John Vugts & Jeroen Brouwer, Philips	UHAPI tutorial	26-2:00-C x2
Karim Yaghmour, Opersys	Building Embedded Linux Systems - Tutorial	26-2:00-A x3
Pieter van der Meulen, Philips	Audio, Video, Graphics WG discussion	26-4:00-C
Dongjun Shin, Samsung	Flash Memory WG discussion	26-4:00-B
Scott Preece, Motorola	Mobile Phone Profile WG - intro and working session	26-12:00-A
Mark Orvek, Monta Vista	Power Management WG discussion - suspend to flash technology, etc.	26-11:00-B
Manas Saksena, Time Sys	Real Time WG discussion	26-2:00-B
Hiroo Suyama, NEC	System Size WG discussion	26-9:00-A

Matsubara-san & Hagiwara-san & Hisao Munakata, Renesas	Graphics APIs for Linux (including DirectFB and 3D)	26-10:00-C
Nigel Cunningham	Linux 2.6 Power Management	26-9:00-B
Emmanuel Fleury & Kristian S�rensen, Aalborg University	Umbrella Security Framework	26-11:00-C
Dennis Oliver Kropp	DirectFB	25-3:00-C
Michael Hunold	Linux DVB	26-12:00-C
Markku Ursin, Movial	Creating GTK+ based UI's for embedded devices	26-11:00-C
John Cooper, Time Sys	2.6 Realtime preemption patch	26-3:00-B
Seiji Munetoh, IBM & Nichola Szeto, Sony	Integrating TCG (Trusted Computing Group) technology in Linux	25-2:00-B
Tim Bird, Sony	Bootup Time WG discussion - current bootup time projects	26-12:00-B
Andrew Morton, Lead kernel developerHisao Munakata, RenesasMark Orvek, Monta VistaRuud Derwig, PhilipsModerated by Tim Bird, Sony	Panel - Improving Commercial CE Involvement in Linux	25-4:00-ABC

Session Schedule

CELF Technical Conference - January 25

1:00-2:00 - Lunch

Time

Room A - Hayes Ballroom

Room B - Monterey room

Room C - Morgan Hill room

2:00 - 2:50 pm

uClinux - Greg Ungerer, SnapGear

Integrating TCG (Trusted Computing Group) technology in Linux - Seiji Munetoh, IBM & Nicholas Szeto, Sony

Linux Kernel State Tracer (LKST) technology - Tohru Nojiri, Hitachi

3:00 - 3:50 pm

Linux on Arm Cores - Philippe Robin, ARM

Umbrella Security Framework - Emmanuel Fleury

DirectFB - Dennis Oliver Kropp

4:00 - 4:50 pm

Panel - Improving commercial CE involvement in Linux - Andrew Morton, Hisao Munakata, Mark Orvek, Ruud Derwig : Moderated by Tim Bird

5:00-6:30 - Demonstrations

7:00 - Dinner

.

CELF Technical Conference - January 26

Time

Room A - Hayes Ballroom

Room B - Monterey room

Room C - Morgan Hill room

9:00 - 9:50

System Size WG discussion - Hiroo Suyama

Linux 2.6 Power Management - Nigel Cunningham

OpenGL ES, Open VF and Open MAX - Ed Plowman - ARM

10:00 - 10:50

uClibc - Erik Andersen

Dynamic Power Management - Todd Poynor, Monta Vista

Graphics APIS for Linux - Matsubara, Hagiwara, Hisao Munakata, Renesas

11:00 - 11:50

Linux-tiny - Matt Mackall

Power Management WG discussion - suspend to flash technology, etc. - Mark Orvek, Monta Vista

Creating GTK+ based UI's for embedded devices - Markku Ursin, Movial

12:00 - 12:50

Mobile Phone Profile WG - intro and working session - Scott Preece

Bootup Time WG discussion - current projects - Tim Bird, Sony

Linux DVB - Michael Hunold

1:00-2:00 - Lunch Break

2:00 - 2:50 pm

Embedding Linux tutorial - Karim Yaghmour

Real Time WG discussion - Manas Saksena, Time Sys

UHAPI tutorial - John Vugts - Philips

3:00 - 3:50 pm

Embedding Linux tutorial2 - Karim Yaghmour

2.6 Real-time preemption patch - John Cooper, Time Sys

UHAPI tutorial2 - John Vugts - Philips

4:00 - 4:50 pm

Embedding Linux tutorial3 - Karim Yaghmour

Flash Memory WG discussion - Dongjun Shin, Samsung

Audio, Video, Graphics WG discussion - Pieter van der Muelen, Philips

5:00-6:00 - Break

6:00 - Dinner

Hotel Information

Name:	Dolce Hayes Mansion
Address:	200 Edenvale Avenue, San Jose, CA 95136, USA
Telephone:	+1 (408) 226-3200
Fax:	+1 (408) 362-2329
Web-site:	http://HayesMansion.Dolce.com
Room rate:	$145.90 per person (single occupancy) per night including breakfast and all taxes.
Reservation:	For hotel reservation please fill in and fax this [[[Media:CELF-Plenary-Hotel-Reservation-Form.pdf]] form (Adobe PDF)] directly to the hotel at +1 (408) 362-2329 by January 3rd, 2005
Map:	MapQuest map

Category:

- Events

> From: eLinux.org

Glossary

This is a glossary of terms used in embedded Linux, and links to existing glossary pages:

Topic-specific Glossaries

Here are pages that have list of terms for specific technology areas:

- Boot-up Time Definition Of Terms
  - terms related to the Linux boot-up process
- Power Management Definition Of Terms
  - Definition of Terms for the CELF Power Management working group
- Real Time Terms - terms related to systems with real-time performance
- Security Terms - terms related to Linux security and security frameworks

Contents

A

Abatron Abatron is a Swiss company that makes a popular Jtag debugger often used to debug embedded Linux. Their primary products are the 'BDIx000' line of Jtag debuggers.

Asynchronous I/O I/O where control is returned to the calling program after the I/O process has started, but before the I/O is completed. The I/O transfer runs in parallel with respect to the processor work. The user program continues executing at the same time the I/O operation is executing.

B

Board A board is used to refer to the hardware upon which one develops embedded Linux. Historically, it refers to the printed circuit board that actually holds the hardware for the device. Often, this is a development board, or evaluation board, as opposed to an actual product device in final shipping form.

Blocking I/O I/O where control is not returned to the calling program until all requested data is transferred. The I/O transfer runs serially with respect to the processor work.

BSP Board Support Package. This is the code that is used to support a particular hardware board. The term is used generically to refer to code, and not a specific "package", as the name implies. It generally refers to all special-case software relevant to a particular board, whether it is kernel code, user code, etc.

C

Cross-compiler A compiler that runs on one platform and has the ability (via configuration) to generate code for a different platform or platforms.

Cross-compilation Compiling code with a cross-compiler in order to target a different platform than the one it's compiled on.

D

E

ELC Embedded Linux Conference. This is one of the major technical conferences each year for developers of embedded Linux. See the Events page for references to past events.

Embedded An "embedded" device usually means one with fixed functionality, independent of other additional hardware or software attributes. The terms is somewhat vague. This is as opposed to general-purpose functionality. Note that mobile phones are considered embedded, although they now support general-purpose programs and functionality. Classic embedded Linux products include things like digital cameras, routers, television sets, and settop boxes, as well as non-consumer embedded devices like sensors, industrial control devices, and pretty much anything running Linux outside of the desktop and server markets. See the wikipedia entry for embedded system.

F

File system The methods and data structures that an operating system uses to keep track of files on a disk or partition; the way the files are organized on the disk. Also used about a partition or disk that is used to store the files or the type of the file system.

G

H

Host The host, or 'host machine' is the place where the software developer actually writes and compiles the software for their product. In a host-target environment, the host is used to develop software, and the target is the device which will run the software.

I

IP block An IP block is a section of an integrated circuit which performs a distinct function on the chip. The IP stands for "Intellectual Property". IP blocks are developed or licensed, and integrated onto an SOC or some other chip in the system. The block is licensed and manipulated as a unit, with the actual definition of the circuity expressed in some hardware description language like Verilog. Because the same IP block may be used in multiple chips (and often in chips even from different companies), a driver written for the IP block on one system may also work (with small modifications) on the same IP block on another system. Common IP blocks on current processors are video controllers, UARTs (serial ports), bus controllers, and network circuitry (both wired and wireless), to name just a few.

J

JTAG Short for "Joint Test Action Group", JTAG refers to a debugging interface used to validate hardware and debug software on an embedded board. See JTAG

K

Kernel The Linux kernel is the core software in a Linux system that interfaces with the hardware, manages resources on behalf of processes, and mediates interactions between processes and the hardware, and between processes themselves.

L

M

N

Non-blocking I/O I/O where control is returned to the calling program after the I/O process has started, but before the I/O is completed. The I/O transfer runs in parallel with respect to the processor work. The user program continues executing at the same time the I/O operation is executing.

Non-volatile storage (NVS, persistent storage, memory) A term describing a storage device whose contents are preserved when its power is off. Storage using magnetic media (e.g. magnetic disks, magnetic tape or bubble memory) is normally non-volatile by nature whereas semiconductor memories (static RAM and especially dynamic RAM) are normally volatile but can be made into non-volatile storage by having a (rechargeable) battery permanently connected.

O

P

PHY Short for Physical Layer, PHY usually refers to the hardware circuitry which implements networking on a chip or board. Sometimes the PHY is a separate chip on the board. Often it refers to the circuitry on an SOC for a particular network device or interface. See http://en.wikipedia.org/wiki/PHY_(chip))

Q

R

RAM-based file system A file system built on RAM as the storage medium.

S

SOC System On Chip. This is a chip which has a (more-or-less) complete system on a single integrated circuit. It will likely have lots of IP blocks, which implement different hardware functionalities, such as serial ports, network interfaces, buses, and video controllers, in addition to the main CPU for the system. It is pronounced as either "ess-oh-see" or like "sock". See http://en.wikipedia.org/wiki/System_on_a_chip

Synchronous I/O I/O where control is not returned to the calling program until all requested data is transferred. The I/O transfer runs serially with respect to the processor work.

T

Target Target refers to the device or environment for which software is being developed. This can be a development board, an actual product, or an emulator. Often, software is developed on a 'host' and then transferred to a target device for testing, debugging and deployment.

Toolchain A toolchain is the set of programs used to build the software for an embedded device. Specifically, it refers to the compiler and linker. But it may also refer to other programs (such as debuggers, profilers, and other tools for working with the target software) that are specific to a particular process architecture or CPU.

U

V

W

X

Y

Z

Category:

- NeedsEditing

> From: eLinux.org

JTAG

Joint Test Action Group, referring to IEEE Standard 1149.1: Four-pin (plus power/ground) interface designed to test connections between chips. Interface is serial (clocked via the TCK pin). Configuration is performed by manipulating a state machine one bit at a time (via TMS pin), then transferring one bit of data in and out per TCK clock (via TDI and TDO pins, respectively). Different instruction modes can be loaded to read the chip ID, sample input pins, drive (or float) output pins, manipulate chip functions, or bypass (pipe TDI to TDO to logically shorten chains of multiple chips). Operating frequency varies per chip, but is typically 10-100MHz TCK (10-100ns per bit time). The description of how JTAG is implemented for a specific device is described in a file specific to that device called a BSDL file.

Contents

Information

- http://en.wikipedia.org/wiki/JTAG
- http://www.asset-intertech.com/products/free_resources.htm
    - various resources
- openjtag wiki
- Common JTAG Pinouts
- Macraigor JSCAN
- Overview of On Chip Debug Methods
- gojtag - needs to be ported to Linux

Tutorial

Tutorial from Asset Intertech

Embedded Linux JTAG debugging (CELF presentation)

Hardware (emulators)

- Holly Gates wiggler - Parallel Port
- Cheap Tag 5v only parallel port
- Flyswatter- USB
- ARM DS-5 - DS-5 and DSTREAM provide embedded Linux developers with a full professional development suite
- gnICE
    - USB w/full schematics
- ICEbear JTAG - USB
- JTAG Blue - Parallel Port
- http://wiki.openwrt.org/JTAG_Cables
- http://hri.sourceforge.net/tools/
    - Macraigor compatible wiggler
- http://www.sparkfun.com/shop/index.php?shop=1&cart=201362&cat=74&
- http://www.ocdemon.net/hwproducts.htm
    - $150-$1800 par,usb,serial,ethernet

- http://www.dlpdesign.com/usb/2232m.shtml
  - USB module
- http://www.corelis.com/products/index.htm
  - Emulators, Testing and Programming
- http://www.amontec.com/jtagkey.shtml
  - USB
- http://www.sparkfun.com/commerce/product_info.php?products_id=7834
  - USB
- http://www.ronetix.at/peedi.html
  - JTAG/BDM Emulator and Flash Programmer
- http://www.mentor.com/products/embedded_software/majic-jtag-probe/
  - JTAG Emulator, Flash Programmer, Hardware Bring-up
- Embest UNetICE - USB and Ethernet based JTAG Emualtor for ARM7 and ARM9 processors
- XDS100v2 - USB JTAG emulator for TI processor available from Embest
- Dangerous Prototypes Bus Blaster (~35$ open hardware)

Tools

- JTAG_Finder
- UsingJtagTools
- http://openwince.sourceforge.net/jtag/
- http://fjtag.sourceforge.net/
- http://jtagtools.sourceforge.net/
- http://www.lart.tudelft.nl/projects/jtag/jflash-linux.tar.gz
- http://lapwww.epfl.ch/dev/arm/jelie/index.php
- http://cvs.arm.linux.org.uk/cgi/viewcvs.cgi/jtag/
- http://jtager.sourceforge.net/
- http://openocd.berlios.de/web/
- http://urjtag.org/
- http://opencollector.org database contains a list of opensource JTAG SW tools

Category:

- Development Tools

From: eLinux.org

Power Management Definition Of Terms

Definition of Terms for Power Management working group

Classes In the future, a Class will be a set of Operating Points. DPM will be architected to choose the appropriate Operating Point for the current Class, based on constraints. The current implementation of DPM restricts each Class to contain only one Operating Point. Continuing the above example of device constraints for a network adapter, if a Class contained two Operating Points, "slow_cpu_fast_io" which specified an IO bus frequency of 25 MHz and "slow_cpu_slow_io" which specified an IO bus frequency of 5 MHz, then DPM would not be able to set the Operating Point to "slow_cpu_slow_io" while the network adapter was active.

Device constraints Constraints describe the required relationships between the hardware operating parameters. For example, for a network adapter to function properly, it might be required that the frequency of the IO bus to which it is attached must be greater than 10 MHz.

DPM The Monta Vista Dynamic Power Management subsystem

Operating Points An Operating Point is a specific set of values of power parameters. The system designer chooses an Operating Point to reflect a certain level of system performance together with an associated level of energy consumption.

Hibernate See Suspend-to-Disk

Operating States The Operating State reflects the current state of the system: executing a task, processing interrupts, or idle. Each process has an associated Operating State, as does the system idle loop and the interrupt handling code path. It is also an abstraction that software uses to request a specific level of system performance, since there are multiple operating states that can be associated with a task, and the various operating states may be mapped to differing Operating Points, thus causing differing levels of performance and power consumption for different tasks. The current system Operating State is set to the Operating State of the process currently executing (or the idle loop or interrupt handlers). The board specific implementation portion of DPM may also alter the system's current Operating State while kernel code is executing. For example, the kernel idle loop code may explicitly set the system's current Operating State to "idle" while there is no task that is ready to execute.

Policy Manager Software that dynamically sets the current policy from the set of defined policies. The Policy Manager selects the current Policy based on certain system conditions. For example, the Policy Manager can select the Current Policy in one of the following ways: 1. If the user presses the device's soft-power switch, the Policy Manager can choose an ultra_low_power Class; 2. If remaining battery power reaches a threshold, the Policy Manager may choose a lower_power_and_performance Class; 3. If the device is connected to main power, the Policy Manager may choose a high_performance Class.

Power Control Layer Power control layer is the software layer on top of which power management framework sits. This layer provides H/W specific power control methods to power management framework with abstracted power control interfaces.

Power Management Framework Power management framework in kernel maintains system power management policy and implements core functions for static power management and dynamic power management technologies. PM framework plays the role of coordinating energy resource among multiple tasks and adapting system operating state according to task specific requirements.

Power Parameters Power parameters are the hardware parameters that may be modified to affect the power usage of the platform, such as clock frequencies or dividers, voltage levels, and so forth.

Power Policy A Policy is a mapping of which Class is invoked for each Operating State. For example, a "power-off" policy may map all Operating States to an extremely low power Class; a "power-off-fast-wakeup" may map all Operating States to a low power and low wakeup latency Class; and a "active" policy may map an Operating State used by interactive processes to a high power Class, an Operating State used by background processes to a medium power Class, and an

Operating State used by the kernel idle loop to a low power and low wake up latency Class. Whenever the Operating State changes (due to an explicit request or implicitly due to a change of which process is currently executing) DPM uses the current Policy to determine which Class is associated with the new Operating State. DPM selects the best Operating Point from the new Class, and then sets the hardware parameters defined by the Operating Point.

Suspend/Resume System suspend refers to placing the system in a low-power state with many components powered off, or perhaps entirely powering off the system after saving state to stable storage. This is a relatively lengthy procedure undertaken when an extended period of product inactivity is expected, as for example when the user presses a "standby" button on the product or the product has not been in use for a long period of time. This procedure can thus be distinguished from the power state changes appropriate for a running system driven by technologies such as Dynamic Power Management, which are executed more quickly during very brief idle periods. System resume refers to the process of restoring the state of the system to approximate pre-suspend conditions upon resume.

Suspend-to-Disk/Hibernate A technique by which systems preserve state on disk during suspend and restore system state from disk upon resume. This is typically done on platforms where all of RAM may be saved to a disk device and then the RAM (and perhaps the entire system) may be powered off during the suspend period. This is commonly referred to as "hibernating" for laptop/notebook computers.

Suspend-to-RAM A technique by which systems preserve state in RAM during suspend and restore system state from RAM upon resume. This is necessary on platforms that remove power from some portions of the system, such as the CPU core or I/O core, but maintain power to SDRAM during the suspend period

Suspend modes / Standby modes / Sleep modes A platform may support more than one suspend mode, that is, system behavior during a suspend may differ depending on what suspend mode has been requested. For example, one mode may stop certain clocks during suspend but leave all components powered on, while another mode may power off various components (such as the CPU core or I/O modules) but leave others powered on (such as to leave SDRAM in self-refresh state), and so forth.

Category:

- Embedded Dictionary

> From: eLinux.org

Real Time Terms

The following terms were copied from the Terminology section of the Realtime Technologies Specification of the CELF.

See http://tree.celinuxforum.org/CelfPubWiki/RTSpecDraft_5fR2

Context switch The act of replacing the running task *A* by another task *B*. The state of task *A* is saved, and *A* is placed in the ready queue; the state of task *B* is restored, and *B* becomes a running task.

Critical section A brief interval during which a task accesses shared data. During this interval, no other tasks are allowed to access the same data. One way of ensuring this is to prevent preemption during the critical section. If the critical section defers preemption for a bounded interval, the resulting priority inversion is bounded.

Deadline-monotonic priority setting The task with the shortest relative deadline is assigned the highest priority.

Deadline requirement A real-time requirement that requires the response to be completed within a fixed time interval from the triggering event. A *relative deadline* is the duration of the above mentioned interval; an *absolute deadline* is the moment in time at which the response must be completed.

Fixed priority preemptive scheduling The task with the highest priority is always running. If a task with a higher priority becomes runnable, the running task will be preempted immediately. The priority of a task, process or Interrupt Service Routine (ISR) is explicitly determined at creation, or by an explicit set-priority command. No implicit priority changes by the scheduler are assumed. For an exception to this rule see **Priority inheritance.** The scheduling policy of SCHED_FIFO in Linux implements fixed priority preemptive scheduling.

```
Note: Fixed priority scheduling is typically designed to be used for a single coherent application.
```

Granularity The time range of a certain interval. We can talk about the granularity of a timing requirement, of a non-interruptible code segment, etc. *Needs fixing*

Hard deadline requirement Missing the deadline is considered an error.

Hard real-time system System with hard real-time requirements.

Hybrid real-time system This refers to a system where a real-time kernel is used in conjunction with a (mostly) unmodified Linux kernel, to provide real-time services to a subset of tasks on the system. This is often also referred to as the "multi-kernel" approach.

Interrupt latency Time passed between interrupt occurrence and activation of interrupt handler.

Interrupt masking Making certain interrupts invisible to the software.

Interrupt response time (worst-case) (Worst-case) time passed between interrupt occurrence and either completion of interrupt service routine (ISR) or wake up of dependent task.

Jitter – absolute Deviation of the occurrence of an event (e.g. completion of frame) from expected occurrence.

Jitter – relative Deviation of the interval between two successive occurrences of an event (e.g. completion of frame) from expected interval.

Mutual exclusion Prevent multiple tasks or ISRs from accessing the same data concurrently. Mutual exclusion is used to protect the integrity of the data.

Preemption A running thread or process can be temporarily suspended. The state of the thread or process (including e.g., program counter, and register values) is saved. When the process or thread is later resumed, the saved state is restored.

Priority inheritance If a high priority-task blocks for a critical section, a low-priority task that holds the lock for the section gets a priority boost. It inherits the priority of the blocked task. This prevents the unbounded priority inversion that could occur if medium priority tasks preempt the lower-priority task that holds the lock. Note that, with priority inheritance, the medium-priority tasks suffer priority inversion as well. But, and this is important in real-time systems, the priority inversion for all tasks is bounded (can be determined without knowing the exact run-time schedule)

Priority inversion The highest priority task is not running. There can be several reasons for priority inversion. One of them is the absence of full preemption. Priority inversion is one of the main reasons for deadlines being missed.

Real-time requirement A requirement on the completion time of a response, generally measured relative to the event that triggered the response.

Real-time system System with one or more real-time requirements.

Response time (worst-case) (worst-case) Time passed between event occurrence and completion of the response to that event. The event may be an interrupt. The response typically involves an interrupt handler and one or more synchronized tasks.

Runnable task (also ready/active task) A task that can run from a logical perspective, but is prevented from running physically.

Semaphore A synchronization primitive often used to achieve mutual exclusion.

Soft deadline Missing deadlines is sometimes acceptable. Compared to hard deadlines, where there is no reason to consider the value of a late result, the value of a late result for a soft deadline is of interest. The value of the result may, for instance, decrease linearly after the deadline.

Soft real-time requirement Soft deadline, or average-case response time requirement.

Note that hard and soft real-time requirements are orthogonal to the temporal granularity that is required. Meeting a soft requirement in the microsecond domain may be more difficult than meeting a hard requirement in the milliseconds domain.

Soft real-time system System with soft real-time requirements.

Contents

Timing Model

It is relatively well-established that the following events and time periods are of interest when measuring realtime performance:

Events

- 0' - event initiation
- A - hardware interrupt assertion
  - This is the time when the hardware interrupt line for an event is raised
- B - Interrupt service routine starts execution
- B' - Task is scheduled
- C - Task starts execution
- D - Result is received from processing the event

Time Periods

Real time performance is often expressed in terms of the maximum, minimum and average duration for certain time periods defined by the events above.

Here are some common terms, expressed relative to those events:

Interrupt latency The time from A to B (from hardware interrupt assertion to ISR start).

Scheduling latency The time from B' to C (from time of task scheduling to process start).

```
Note that sometimes this term is used to refer to the time from A to C (from interrupt assertion to process start).
```

Processing time The time from C to D (from process start to completion of processing).

Response time The time from A to D (total time from interrupt assertion to delivery of processed data).

Often, these or similar terms are used with less accuracy to describe the closest approximation one can get, with a particular test framework and instrumentation set.

For example, 'response latency' may be reported as "response time", and refer to the time from when a host program records the time, previous to transmitting a piece of data which will cause an interrupt on the target machine, to the time when a host program records the time after receiving some signal from the target that processing is completed. Of course, time is taken in the operations of recording time, transmiting data, and detecting signals on the host machine. But it may be that timing these individual operations at a finer granularity requires instrumentation or hardware support not reasonably available for a test.

Terms in need of definition

- Reservation system
- Precision
- Granularity (above definition is weak)

Category:

- Linux

> From: eLinux.org

Security Terms

Linux Security Modules (LSM) A framework to support security systems as loadable Linux modules.

Stack guarding A mechanism for protecting the system from buffer overrun ("stack smashing") attacks.

Category:

- Security

Toolbox

This chapter has information about developing Embedded Linux, including links to toolchains, debuggers and other development tools. Also, it has links to pages with debugging tips.

Development Tools

Tools from hardware logic analyzers, build system, emulators, benchmarks, test systems to software debuggers.

> From: eLinux.org

Logic Analyzers

A logic analyzer is an electronic instrument that displays signals in a digital circuit that are too fast to be observed and presents it to a user so that the user can more easily check correct operation of the digital system. They are typically used for capturing data in systems that have too many channels to be examined with an oscilloscope. Software running on the logic analyzer can convert the captured data into timing diagrams, protocol decodes, state machine traces, assembly language, or correlate assembly with source-level software.

When selecting a logic analyzer, make sure that the software package includes bus analyzers (I2C/SPI/UART are a given). Any tool worth purchasing will include support for these.

Contents

Tools

Since logic analyzers typically need software running on your development system, this list is split up to give preference to those tools which are fully supported on both Windows and Linux.

Linux

- BitScope:
  - 100Mhz
  - Cheapest one looks to be around 475 USD
- miniLA - open source hardware!
- Saleae:
  - 8 Inputs
  - USB 2.0
  - 24MHz max
  - Logic analyzer with RS232, SPI, I2C and 1-Wire Protocol Analyzer
  - Can handle 1.8V logic (Spec mentions *Input High Voltage: 2 to 5.25V*, but users report that 1.8V work, too)
  - ~149 USD
- Sigma2
  - 16 inputs
  - 200MHz
  - €198

Everyone Else

- Acute
- ASIX
- TechTools
- LogicPort:

    - Windows only SW
    - up to 500Mhz?
    - ~389 USD
- Zeroplus (e.g. LAP-C):
    - Can handle 1.8V logic
    - Can do decodes of the data
    - Around 100USD depending of the device.
    - Some devices seems to be available with Linux software (?)

Information

- http://en.wikipedia.org/wiki/Bus_analyzer
- http://en.wikipedia.org/wiki/Logic_analyzer

Category:

- Development Tools

> From: eLinux.org

Toolchains

A toolchain is a set of distinct software development tools that are linked (or chained) together by specific stages such as GCC, binutils and glibc (a portion of the GNU Toolchain). Optionally, a toolchain may contain other tools such as a Debugger or a Compiler for a specific programming language, such as ,C++. Quite often, the toolchain used for embedded development is a cross toolchain, or more commonly known as a cross compiler. All the programs (like GCC) run on a host system of a specific architecture (such as x86) but produce binary code (executables) to run on a different architecture (e.g. ARM). This is called cross compilation and is the typical way of building embedded software. It is possible to compile natively, running gcc on your target. Before searching for a prebuilt toolchain or building your own, it's worth checking to see if one is included with your target hardware's Board Support Package (BSP) if you have one.

Contents

Introduction

When talking about toolchains, one must distinguish three different machines :

- the build machine, on which the toolchain is built
- the host machine, on which the toolchain is executed
- the target machine, for which the toolchain generates code

From these three different machines, we distinguish four different types of toolchain building processes :

- A native toolchain, as can be found in normal Linux distributions, has usually been compiled on x86, runs on x86 and generates code for x86.
- A cross-compilation toolchain, which is the most interesting toolchain type for embedded development, is typically compiled on x86, runs on x86 and generates code for the target architecture (be it ARM, MIPS, PowerPC or any other architecture supported by the different toolchain components)
- A cross-native toolchain, is a toolchain that has been built on x86, but runs on your target architecture and generates code for your target architecture. It's typically needed when you want a native gcc on your target platform, without building it on your target platform.
- A canadian build is the process of building a toolchain on machine A, so that it runs on machine B and generates code for machine C. It's usually not really necessary.

Toolchain components

Binutils

The GNU Binutils are the first component of a toolchain. The GNU Binutils contains two very important tools :

- *as*, the assembler, that turns assembly code (generated by gcc) to binary
- *ld*, the linker, that links several object code into a library, or an executable

Binutils also contains a couple of other binary file manipulation or analysis tools, such as objcopy, objdump, nm, readelf, strip, and so on. The Binutils website has some documentation on all these tools.

C, C++, Java, Ada, Fortran, Objective-C compiler

The second major component of a toolchain is the compiler. In the embedded Linux, the only realistic solution today is GCC, the GNU Compiler Collection. Nowadays, as input, it not only supports C, but also C++, Java, Fortran, Objective-C and Ada. As output, it supports a very wide range of architectures.

C library

The C library implements the traditional POSIX API that can be used to develop userspace applications. It interfaces with the kernel through system calls, and provides higher-level services.

Realistically, there are nowadays two options for the C Library:

- glibc is the C library from the GNU project. It's the C library used by virtually all desktop and server GNU/Linux systems. It's feature-full, portable, complies to standards, but a bit bloated.
- Embedded GLIBC (EGLIBC) is a variant of the GNU C Library (GLIBC) optimized for embedded systems. Its goals include reduced footprint, support for cross-compiling and cross-testing, while maintaining source and binary compatibility with GLIBC. The project is discontinued.
- uClibc is an alternate C library, which features a much smaller footprint. This library can be an interesting alternative if flash space and/or memory footprint is an issue. However, the space advantages gained using uClibc are becoming less important as the price of memory & flash continues to drop. It is still useful C library for embedded systems without MMU.
- uClibc-ng is a spin-off of uClibc C library. Main goal of the spin-off is to do regular releases and do a lot of automatic runtime testing.
- musl New standard C library. musl is lightweight, fast, simple, free, and strives to be correct in the sense of standards-conformance and safety.

The C library has a special relation with the C compiler, so the choice of the C library *must* be done when the toolchain is generated. Once the toolchain has been built, it is no longer possible to switch to another library.

Debugger

The debugger is also usually part of the toolchain, as a cross-debugger is needed to debug applications running on your target machine. In the embedded Linux world, the typical debugger is GDB.

Lazarus and Free Pascal

Free Pascal is a professional but free 32 bit / 64 bit compiler for Pascal and ObjectPascal. It supports a wide variety of processors and Linux distributions including the Raspberry Pi.

The Free Pascal toolchain is widely independent from GCC and other external tools. Major components are the Free Pascal compiler (FPC), a command-line tool, a text-mode IDE and, as an optional component, Lazarus, a full-featured GUI-based IDE. FPCUnit is a framework allowing for unit-testing.

On most platforms Free Pascal makes use of the GDB debugger.

http://elinux.org/Tiny6410

http://elinux.org/Micro2440

http://elinux.org/Mini210

http://elinux.org/Tiny210

Getting a toolchain

There are several ways to get a toolchain :

- Get a prebuilt toolchain, either from a vendor such as CodeSourcery, or probably inside the Board Support Package shipped with your hardware platform by the vendor. This is the easiest solution, as the toolchain is already built, and has supposedly been tested by the vendor. The drawback is that you don't have flexibility on your toolchain features (which C library ? hard-float or soft-float ? which ABI ?)
- Build a toolchain on your own. However, this can be a real pain. There are version dependency issues, patches required to make something work etc. etc. Check out this (obsolete) build matrix for crosstool and look at all the red "failed" entries.
- Build a toolchain using an automated tool. The community has built several scripts or more elaborate systems to ease the process of building a toolchain. This way, the recipes and patches needed to build a toolchain made of particular versions of the various components are shared and easily available.

Prebuilt toolchains

CodeSourcery

CodeSourcery develops Sourcery G++, an Eclipse based Integrated Development Environment (IDE) that incorporates the GNU Toolchain (gcc, gdb, etc.) for cross development for numerous target architectures. CodeSourcery provides a "lite" version for ARM, Coldfire, MIPS, SuperH and Power architectures. The toolchains are always very up to date. CodeSourcery contributes enhancements it makes to the GNU Toolchain upstream continually, making it the single largest (by patch count) corporate contributor.

Linaro (ARM)

Linaro releases optimized toolchains for recent ARM CPUs (Cortex A8, A9...). These include Linaro's latest contributions to mainline gcc, but backported to stable gcc versions for immediate use by product developers. Linaro actually hires CodeSourcery people to improve ARM toolchains, so the ARM toolchains that you get with Linaro should be at least as good as the CodeSourcery ones.

Native toolchains are available through the standard gcc toolchain in Ubuntu. Cross toolchains are available to Ubuntu users through special packages:

```
sudo add-apt-repository ppa:linaro-maintainers/toolchain
sudo apt-get install gcc-arm-linux-gnueabi
```

Now find out the path and name of the cross-compiler executable by looking at the contents of the package:

```
dpkg -L gcc-arm-linux-gnueabi
```

Arch Linux users can install:

```
yaourt -S gcc-linaro-arm-linux-gnueabihf
```

Linaro also makes source releases which can then be used by any build system (see below).

DENX ELDK

The DENX Embedded Linux Development Kit (ELDK) provides a complete and powerful software development environment for embedded and real-time systems. It is available for ARM, PowerPC and MIPS processors and consists of:

- Cross Development Tools (Compiler, Assembler, Linker etc.) to develop software for the target system.
- Native Tools (Shell, commands and libraries) which provide a standard Linux development environment that runs on the target system.
- Firmware (U-Boot) that can be easily ported to new boards and processors.
- Linux kernel including the complete source-code with all device drivers, board-support functions etc.
- Xenomai - RTOS Emulation framework for systems requiring hard real-time responses.
- SELF (Simple Embedded Linux Framework) as fundament to build your embedded systems on.

All components of the ELDK are available for free with complete source code under GPL and other Free Software Licenses. Also, detailed instructions to rebuild all the tools and packages from scratch are included.

The ELDK can be downloaded for free from several mirror sites or ordered on CD-ROM for a nominal charge (99 Euro). To order the CD please contact office@denx.de

Detailed information about the ELDK is available here.

Scratchbox

Scratchbox provides toolchains for ARM and x86 target architectures (with PowerPC, MIPS and CRIS in experimental stages but aren't making real progress since years, so Scratchbox should probably be considered ARM and x86 only). Both uClibc and glibc are supported.

Scratchbox simplifies cross compiling software which is built using GNU autotools - Code tests performed by configure are run in an emulator or even on the actual target. The toolchains scratchbox ships with are based on gcc 3.3 and as such are quite old, but stable and well tested. It should be pointed out that scripts to build custom toolchains are also provided with scratchbox allowing more recent gcc versions to be used.

Fedora ARM

Fedora ARM is a try to port Fedora to ARM. It provides some tools as an ARM toolchain packaged in RPM format. Link: Fedora ARM

Debian cross-tools packages

For Debian users, the toolchains problem is fairly reliably solved.

For a debian-based box just install pre-built cross toolchains from Debian experimental.

Targets include nearly all debian-supported architectures As of this writing supported compiler is gcc 4.9. You can get older unsupported compilers from emdebian.

You will need to add the target architecture to your list of installable architectures. eg.

```
dpkg --add-architecture armhf
apt-get update
apt-get install gcc-arm-linux-gnueabihf
```

Free Pascal

Free Pascal is available for several processor architectures from http://www.freepascal.org, the Lazarus IDE from http://www.lazarus.freepascal.org.

Toolchain building systems

Buildroot

Buildroot is a complete build system based on the Linux Kernel configuration system and supports a wide range of target architectures. It generates root file system images ready to be written to flash. In addition to having a huge number of packages which can be compiled into the image, it also generates a cross toolchain to build those packages from source. Even if you don't want to use buildroot for your root filesystem, it is a useful tool for generating a toolchain. Buildroot supports uClibc, glibc, eglibc, and musl.

OpenADK

OpenADK is a complete build system based on the Linux Kernel configuration system and supports a wide range of target architectures. It is similar to buildroot. It generates root file system images ready to be written to flash. In addition to having a huge number of packages which can be compiled into the image, it also generates a cross toolchain to build those packages from source. Even if you don't want to use OpenADK for your root filesystem, it is a useful tool for generating a toolchain. OpenADK supports uClibc, glibc, uClibc-ng, and musl.

Crossdev (Gentoo)

Crossdev is specific to developers using Gentoo for their development PCs. It is a script which generates a cross toolchain using the portage build scripts for gcc etc. There are numerous architectures which are supported and both uClibc and glibc toolchains can be built. Link: Gentoo Crossdev info

Crosstool-NG

Crosstool-NG is a well-maintained fork of crosstool, targeted at easier configuration, re-factored code, and a learning base on how toolchains are built, with support for both uClibc and glibc, for debug tools (gdb, strace, dmalloc...), and a wide range of versions for each tools. Different target architectures are supported as well. It offers a kernel-like configuration system to select the different configuration options of the toolchain (component versions, component configuration, etc.). Crosstool-NG has an active and responsive user and developer community.

Crossdev/tsrpm (Timesys)

Crossdev is a project sponsored by Timesys, completely unrelated to the Gentoo cross toolchain generation system. The projects main focus is on a tool called tsrpm which is used to build cross development toolchains and generate cross-compiled software packages. Currently only x86 and select PowerPC architectures are supported. Link: Crossdev

EmbToolkit

EmbToolkit is the first toolchain building system giving the choice to generate GCC or llvm/clang based toolchain. *EmbToolkit* supports use of eglibc, glibc or uClibc as C Library and musl C library is also planned at time of writing. *EmbToolkit* can be used to generate only a toolchain (usable in a external project), but it is also possible to generate various root filesystems.

OSELAS.Toolchain()

The OSELAS.Toolchain() project aims at supplying a complete build system for recent GNU toolchains. It uses the PTXdist build system, a userland build system based on Kconfig. The current version 1.99.3.1 of OSELAS.Toolchain() contains support for arm, x86, avr, mips and PowerPC. In addition, there are toolchains for bare metal platforms like Cortex-M3 and AVR-8-Bit.

OSELAS.Toolchain() Feature matrix (v1.99.1, build with PTXdist v1.99.7)

Architeture

CPUtype

gcc

glibc

binutils

kernel header

ARM (eabi)

1136jfs,xscale(-hf),iwmmx,v4t(-hf),v5te,xscale

4.1.2, 4.3.2

2.5, 2.8

2.17, 2.18

2.6.18, 2.6.27

AVR

n/a

3.4.6, 4.1.2, 4.3.2

1.0.5, 1.4.8, 1.6.2

2.17

n/a

X86

i586, i686

4.1.2, 4.3.2

2.5, 2.8

2.17, 2.18

2.6.18, 2.6.27

PowerPC

603e

4.1.2, 4.3.2

2.5, 2.8

2.17, 2.18

2.6.18, 2.6.27

MIPS

mipsel (softfloat)

4.2.3

2.8

2.18

2.6.27

OSELAS.Toolchain() contains some further goodies like gcj support for ARM and mingw Support for x86. Link: OSELAS.Toolchain() Link: PTXdist

Bitbake

Bitbake is the tool used by OpenEmbedded. The best way to get started is probably by just building an existing distribution that uses openembedded (e.g. Ångström, see http://www.angstrom-distribution.org/building-ångström for details).

By Platform

ARM

See ARMCompilers

Category:

- Development Tools

> From: eLinux.org

Build Systems

- Open Embedded - System for building full embedded images from scratch. Note that this is used by the Yocto Project as it's build system (but see note below for more detail).
- Buildroot - Easy-to-use embedded Linux build system
- OpenADK - Open Source Appliance Development Kit
- PTXdist
    - Kconfig based build system developed by Pengutronix
    - GPL licensed
    - Video of a talk given by PTXdist maintainer Robert Schwebel at FOSDEM 2009
- Linux From Scratch
- Nard SDK For industrial embedded systems
- LTIB - Linux Target Image Builder (by Stuart Hughes of FreeScale) - see http://savannah.nongnu.org/projects/ltib
    - Slides and video of a talk on LTIB at the Ottawa Linux Symposium 2008
- OpenBricks
    - Embedded Linux Framework
    - OpenBricks provides a set of packages, patches and shell-based rules that creates a toolchain and a rootfs with customized packages and features selection.
    - Currently supports x86_32, x86_64, PowerPC, PowerPC64 and ARM architectures with either uClibc, Glibc or eGlibc C library.
- Building Embedded Userlands - Presentation by Ned Miljevic & Klaas van Gend at the ELC 2008 which compares different configuration and build systems. Video of the conference available.
- Scratchbox Cross-Compilation Toolkit, with support for x86 and arm.
- OpenWRT Cross-Compilation Toolkit mainly geared towards wireless routers but can be extended to other platforms, with support for x86, MIPS and ARM.
- Yocto Project - The Yocto Project is an umbrella project which uses Open Embedded as it's build system.
    - I have heard it argued that the Poky meta-data for Open Embedded actually constitute the primary build system for the Yocto Project. Since Open Embedded somewhat conflates the package data and the build scripts in the recipe files, there is some truth to this.

See also Toolchains

Category:

- Development Tools

> From: eLinux.org

Embedded Linux Distributions

Contents

Introduction

Besides the Linux kernel, one of the advantage of embedded Linux is the ability to leverage hundreds if not thousands of existing free and open source packages to easily and quickly add new features to devices. These packages range from graphical libraries, multimedia libraries, network libraries, cryptographic libraries, network servers, infrastructure software and more. However, integrating all these components together requires a relatively deep knowledge of the components. Hence, embedded-specific distributions and build systems have been created to ease this process.

Vendor distros

- The Blackfin uClinux Distribution by Analog Devices - a fork of the uClinux distribution for Blackfin processors
- Embedded Alley - see http://www.embeddedalley.com/
- KaeilOS embedded linux
- Lineo Solutions uLinux
- MontaVista Linux - see http://www.mvista.com/products_services.php
- Pengutronix OSELAS.BSP() - see http://www.pengutronix.de/oselas/bsp/index_en.html
- RidgeRun Linux - see http://www.ridgerun.com/sdk.shtml
- TimeSys LinuxLink - see http://www.timesys.com/embedded-linux/linuxlink
- Ubuntu Mobile
- Wind River - see http://www.windriver.com/products/linux/
- Little Blue Linux - MPC Data
- Digi Embedded Linux for Digi's ARM based modules

Platforms

- Android
- Maemo (deprecated - see Meego)
- Meego
- Moblin (deprecated - see Meego)
- OpenMoko
- Access Linux Platform
- LIMO

Other distros

- Snapgear Embedded Linux Distribution - http://www.snapgear.org/
- Open Wrt - http://openwrt.org/
- Embedded Debian - http://www.emdebian.org/
  - Emdebian has made some significant progress the last few years, and has now reached an usable state. Two versions of Emdebian are available : Emdebian Grip and Emdebian Crush.
  - Neil Williams gave a talk *Emdebian 1.0 release, small and super small Debian* at the FOSDEM 2009. A video is available.
- Embedded Gentoo - http://embedded.gentoo.org/
- Arch Linux for ARM - http://archlinuxarm.org/
- GeeXboX - http://www.geexbox.org/
  - GeeXboX is an embedded multimedia distribution that turns your device into a full-featured MediaCenter.
- Aboriginal Linux - http://landley.net/aboriginal/

Special purpose embedded Linux distributions

- Flash Linux - a distribution specifically for USB keys and Live CDs
- Eagle Linux - http://www.safedesksolutions.com/eaglelinux/
  - An embedded Linux distribution aimed at helping users learn Linux by creating bootable Linux images "virtually from scratch". Eagle Linux 2.3 is currently distributed as a concise, 26-page PDF documenting the creation of a minimalist, network-ready Linux image for bootable CDs, floppies, or flash drives. See description at: http://ct.enews.deviceforge.com/rd/cts?d=207-106-2-28-5560-8662-0-0-0-1
- uClinux A distribution targeting (but not only) systems without Memory Management Unit. See also UClinux Shared Library.

Configuration and Build systems

See Build Systems

See also Toolchains

Obsolete things

- Qplus Target Builder
  - Target image builder from ETRI

Further reading

- Embedded OS mentions a variety of embedded operating systems, including embedded Linux.
- Tools and distributions for embedded Linux development - LWN.net 2010/04/27 by Tom Parkin
  - This is an excellent roundup of current (as of 2010) tools and distributions available for embedded Linux development (that's redundant).

Category:

- Linux

> From: eLinux.org

Debuggers

Debugging is one of the most common activities of an embedded developer. Here are some debuggers howto and links:

- Open On-Chip Debugger
  - OpenOCD local pages.
- GDB - The GNU debugger
- DDD - Data Display Debugger
- kgdb - kernel source level debugger
- KDB - In-kernel debugger
- valgrind - memory, cache and other debuggers and profilers

Category:

- Development Tools

> From: eLinux.org

Debug Assist Boards

Here are some debug assist boards that you might find useful.

Contents

List of debug-assist boards for embedded Linux

Jtag boards

- Olimex JTAG USB OCD device - see https://www.sparkfun.com/products/7834
  - has USB JTAG, RS232 (full modem signals supported) port; and power supply all in one compact device
- Abatron BDI3000 - see http://www.abatron.ch/products/bdi-family/bdi3000.html

Control boards

- Sony Debug Assist board (also known as CDB Assist)
  - provides power supply, USB pass-through to host, and control of 3 "buttons" via USB connection on host
- Target Switch Control From Parallel Port
  - an old design for controlling switches using a computer's parallel port lines

Power Measurement

- BayLibre ACME cape (for BeagleBone Black) - see http://baylibre.com/acme/
  - supports different power measurement probes via a standard connector
  - presentation on issues at: http://events.linuxfoundation.org/sites/events/files/slides/Leveraging_Open-Source_Power_Measurement_Standard_Solution_0.pdf

> From: eLinux.org

Memory Debuggers

Several tools exist for finding memory leaks or for reporting individual memory allocations of a program. These tools help analyze memory usage patterns, detect unbalanced allocations and frees, report buffer over- and under-runs, etc.

Contents

mtrace

mtrace is a builtin part of glibc which allows detection of memory leaks caused by unbalanced malloc/free calls. To use it, the program is modified to call mtrace() and muntrace() to start and stop tracing of allocations. A log file is created, which can then be scanned by the 'mtrace' Perl script. The 'mtrace' program lists only unbalanced allocations. If source is available it can show the source line where the problem occurred. mtrace can be used on both C and C++ programs.

See the mtrace wikipedia article for more information.

memwatch

memwatch is a program that not only detects malloc and free errors but also reads and writes beyond the allocated space (buffer over and under-runs). To use it, you modify the source to include the memwatch code, which provides replacements for malloc and free.

Some things that memwatch does not catch are writing to an address that has been freed and reading data from outside the allocated memory.

mpatrol

mpatrol appears to be like memwatch.

See http://mpatrol.sourceforge.net/

dmalloc

"The debug memory allocation or dmalloc library has been designed as a drop in replacement for the system's malloc, realloc, calloc, free and other memory management routines while providing powerful debugging facilities configurable at runtime. These facilities include such things as memory-leak tracking, fence-post write detection, file/line number reporting, and general logging of statistics."

This library can be used without modifying the existing program, and uses environment variables to control it's operation and set of issues to log.

It's home page is at: http://dmalloc.com/

See Cal Erickson's article (link below, page 2) for information about using this system.

dbgmem

dbgmem looks like another dynamic library replacement tool, similar to dmalloc (but possibly having less features)

See http://dbgmem.sourceforge.net/

valgrind

valgrind does dynamic binary instrumentation to analyze the program, and provides a number of memory problem detection tools and profiling tools. Unfortunately, as of July 2010 it is only available for x86 and ppc64 architecture platforms.

See Valgrind

Electric Fence

See Electric Fence

Tutorials or Overviews

- Memory Leak Detection in Embedded Systems by Cal Erickson, Linux Journal, September 2002
  - This article mentions mtrace, memwatch and dmalloc

Category:

- Development Tools

> From: eLinux.org

Tools

- addr2line for kernel debugging
- decode an ioctl
- Exception Analysis tools
  - This includes things like oops emitters, exception monitors, coredump analysis, etc.
- PgpKey
- Ttc - target control tool

> From: eLinux.org

Integrated Development Environments

- Eclipse - Powerful IDE written in JAVA.
- Free Pascal supports both a text-mode IDE and Lazarus (see below)
- jEdit - Editor written in JAVA which can be expanded to a full IDE with plug-ins.
- KDevelop - Standard IDE for KDE.
- Lazarus, a cross-platform IDE for Free Pascal, a 32bit / 64bit professional ObjectPascal compiler.
- Emacs - Powerful IDE, extensible in LISP, ships with modes to integrates with SCM (GIT, SVN, CVS...), build systems, debugger and even fancy multi-window with ECB.
- VIm - Powerful IDE, extensible with scripting, can use various modules for completion and more.
- KScope - Cscope based source editing environment with KDE.
- Anjuta - IDE with nice plugin support
- QtCreator is an nice IDE which has code completion, remote deployment (with version 2.3) and Outline view. It also has an VIm mode. Its menus are much cleaner than these from Eclipse and its easier to get started with this ide than Eclipse for that very reason.
- Embest IDE for ARM

Category:

- Development Tools

> From: eLinux.org

Emulators

- Qemu - hardware emulator - for everything (try this first)
- Skyeye - for ARM
- Aranym - for M68K
- Hercules - For S390
- UNetICE for ARM7 and ARM9 processors from Embest
- XDS100v2 for TI processor available from Embest

> From: eLinux.org

Tracers and Profilers

- Strace - trace system calls by a program (or set of programs) (very handy)
- ltrace
    - trace library calls
- see Kernel Trace Systems
- see Profilers

Category:

- Development Tools

> From: eLinux.org

Benchmark Programs

Here are some different programs for performing benchmarking.

Note: It is important to recognize that benchmarks between systems may be misleading. Benchmarks should primarily be used to determine differences in performance for different software configurations on the **same** hardware system.

Contents

Unix Bench

FYI, the URL to the UnixBench is as follows;

OLD site: http://www.tux.org/pub/tux/benchmarks/System/unixbench/

NEW site: http://code.google.com/p/byte-unixbench/

UnixBench contains 9 kinds of tests:

1. Dhrystone 2 using register variables
2. Double-Precision Whetstone
3. Execl Throughput
4. File Copy
5. Pipe Throughput
6. Pipe-based Context Switching
7. Process Creation
8. Shell Script
9. System Call Overhead

lmbench

The LMBench home page is at: http://www.bitmover.com/lmbench/ and/or http://lmbench.sourceforge.net/ The sourceforge project page is at: http://sourceforge.net/projects/lmbench

Instructions for lmbench-3.0-a9

(Adjust CC and OS according to your needs.)

```
cd lmbench-3.0-a9/src
make CC=arm-linux-gcc OS=arm-linux TARGET=linux
```

Make the whole lmbench-3.0-a9 directory accessible on the target, e.g. by copying or NFS mount. Make sure the benchmark scripts can write the configuration file and results, and also unpack a tarball used during the benchmark (in case tar is not available on target):

```
chmod a+w ../bin/arm-linux ../results
tar xf webpage-lm.tar
```

To run the benchmark on the target:

```
cd lmbench-3.0-a9/src
hostname foo    # make sure hostname is set, the scripts use it to name config and result files
OS=arm-linux ../scripts/config-run
OS=arm-linux ../scripts/results
```

This worked for me on a target using BusyBox v1.10.2 ash.

The results are written into lmbench-3.0-a9/results/, for each run of the ../scripts/results a new file is created. You can copy the results back to your PC and run various kinds of summary postprocessing scripts from lmbench, e.g.

```
../scripts/getsummary ../results/arm-linux/*
```

Wishlist

A list of benchmark results would be useful:

- Comparing performance of different FFT implementations on Beagleboard-XM: http://pmeerw.dyndns.org/blog/programming/arm_fft.html

Category:

- Development Tools

> From: eLinux.org

Source Management Tools

Here are some different source management tools commonly used with Linux:

Contents

Overview

- David Wheeler has an excellent breakdown of various SCM tools at: http://www.dwheeler.com/essays/scm.html
- IBM has an good overview of available tools at: http://www-128.ibm.com/developerworks/linux/library/l-vercon/
- There is a comparison of several different tools at: http://better-scm.berlios.de/comparison/comparison.html

Patch Management Tools

- diff - to create patches
  - use 'man diff' on your local system for information
- patch - to apply patches
  - use 'man patch' on your local system for information
- Quilt is good for managing a group of patches relative to a single source base.
- diffstat reads a patch file (or standard input) and displays a histogram of the insertions, deletions, and modifications per-file. It is useful for reviewing large, complex patch files. It reads from one or more input files or from standard input. If an input filename ends with .bz2, .Z or .gz, diffstat will read the uncompressed data via a pipe from the corresponding program.
  - diffstat is included in most Linux distributions
  - diffstat home page
  - diffstat man page
- Tim's patch management tools
  - diffinfo and friends - a more verbose diffstat, with splitting, joining and comparing of patches
- See also Diff And Patch Tricks
- Splash - Sony tool for managing patches (kind of like quilt or stacked git)

Version Control Systems

- Git
- Subversion
- BitKeeper

Identity Verification, text validation

- PgpKey

Category:

- Development Tools

> From: eLinux.org

Test Systems

Here is a quick list of different test systems (and test projects) for Linux:

Contents

Test Projects

Clearly, a number of open source projects in this realm exist. This page intends to collect descriptions and links to these projects

Test Automation Tools

- Autotest
- LAVA - Linaro's test framework
- Opentest
- Red Hat Test Project
- Ktest Automate kernel testing
- Jenkins-based Test Automation
  - the official test framework for the LTSI project

Papers on testing

- Paper on finding bugs in Unix programs using random input: http://ftp.cs.wisc.edu/paradyn/technical_papers/fuzz.pdf

Test Suites

- Linux Test Project
- LTP-DDT

Static Analysis

- Sparse
- Smatch
- clang

Automated kernel compilation results

Here are some locations where automated tests of kernel compilation can be viewed:

L4X

- http://l4x.org/k/ - Jan Dittmer's page showing the build status and kernel size of the defconfigs of many architectures. Running since 2004 or 2005

Kautobuild

Kautobuild is Simtec's automated system to build and store results for ARM and MIPS platforms, for every kernel version. It uses defconfigs for multiple boards, and reports compile errors/warnings, module size, kernel size etc.

- Kautobuild for ARM - http://armlinux.simtec.co.uk/kautobuild/
- mail archive version of results is at: http://lists.simtec.co.uk/pipermail/kautobuild/
- Article about the Simtel site is at: http://www.linuxdevices.com/news/NS4646877354.html (2006)

ABAT

ABAT - https://sourceforge.net/projects/abat/

```
- does a download/build/boot regression test for several mainline kernel trees, as soon as new versions are available
- test results matrix is available here:
   - http://ftp.kernel.org/pub/linux/kernel/people/mbligh/abat/regression_matrix.html
```

Bug tracking system

A **bug tracking system** is a software application that is designed to help quality assurance and computer programmers keep track of reported software bugs in their work.

Many bug-tracking systems, such as those used by most open source software projects, allow users to enter bug reports directly. Other systems are used only internally in a company or organization doing software development. Typically bug tracking systems are integrated with other software project management applications.

Having a bug tracking system is extremely valuable in software development, and they are used extensively by companies developing software products.

Components

A major component of a bug tracking system is a database that records facts about known bugs. Facts may include the time a bug was reported, its severity, the erroneous program behavior, and details on how to reproduce the bug; as well as the identity of the person who reported it and any programmers who may be working on fixing it.

Typical bug tracking systems support the concept of the life cycle for a bug which is tracked through status assigned to the bug. A bug tracking system should allow administrators to configure permissions based on status, move the bug to another status, or delete the bug. The system should also allow administrators to configure the bug statuses and to what status a bug in a particular status can be moved to.

Usage

In a corporate environment, a bug-tracking system may be used to generate reports on the productivity of programmers at fixing bugs. However, this may sometimes yield inaccurate results because different bugs may have different levels of severity and complexity. The severity of a bug may not be directly related to the complexity of fixing the bug. There may be different opinions among the managers and architects.

A *local bug tracker (LBT)* is usually a computer program used by a team of application support professionals to keep track of issues communicated to software developers. Using an LBT allows support professionals to track bugs in their "own language" and not the "language of the developers." In addition, LBT use allows a team of support professionals to track specific information about users who have called to complain that may not always be needed in the actual development queue (thus, there are two tracking systems when an LBT is in place.)

External links

- Trac - Web-based software project management and bug/issue tracking system.
- Bugzilla - Bug tracking used by the Mozilla projects. Inherently web-based, written in Perl , and uses MySQL as its database back-end. Open-Source.
- Bugreport is a place to publish the security advisories
- How to Report Bugs Effectively

> From: eLinux.org

Test Tools

Test Tools

Tools to aid with Testing

SPI

- spidev_test.c
- SpiSlaveZero SPI Slave Zero universal SPI peripheral

USB

- Gadget Zero

> From: eLinux.org

Scripting

Scripting is powerful technology especially valuable in embbedded Linux. It is used for building complex projects, building root file systems and distributions, system management, tests automation.

Most commons shells are bash on PC and busybox's ash on embedded Linux.

Contents

Shell scripting

- Bash Reference Manual
- Bash Guide for Beginners
- Advanced Bash-Scripting Guide
- Command-Line Tools Summary
- http://wiki.bash-hackers.org/
- http://bash.cyberciti.biz/
- Top 10 Best Cheat Sheets and Tutorials for Linux / UNIX Commands
- http://wiki.bash-hackers.org/
- https://wiki.archlinux.org/index.php/bash
- http://www.techbar.me/linux-shell-tips/

Shell scripting libraries

- shtool The GNU Portable Shell Tool
  - man shtool
  - Portable wrappers to standard operations
  - 3000 SLOC, 19 functions, bloated
- Wicked Cool Shell Scripts, 2004, samples
  - Indeed cool shell scripts, worth to read
  - Most in interesting functions: 015-newrm.sh, 016-unrm.sh, 021-findman.sh, 029-loancalc.sh, 037-zcat.sh, 038-bestcompress.sh, 040-diskhogs.sh, 084-webaccess.sh, 100-hangman.sh
  - 4000 SLOC, 100 files-functions
  - Easy to use
- mbfl - Marco's Bash Functions Library
  - 5000 SLOC, 500 functions, bloated
  - The philosophy of MBFL is to do the work as much as possible without external commands.
  - Complicated to use
- @ aka monkey-tail
  - 300 SLOC, 20 functions, simple wrapper functions
  - Easy to use
- lib.sh

    - Single script, 300 SLOC, 40 functions and aliases
    - Easy to use
    - Most useful functions: make-debug, trap_err, readline-bindings, duplicates, fs_usage, system_status_short, git_fixup, tcpdump-text, git_ign_add, for_each, mem_avail_kb

Specialized frameworks and libraries

- Bashinator
    - Logging framework
    - 700 SLOC, 18 functions
    - Complicated to use
- libbash - tool for managing bash scripts
    - loads and unloads functions from scripts with commands source and unset
- bsfl - Bash Shell Function Library
    - 600 SLOC, 50 functions, logging functions, trivial wrappers
    - Easy to use
- shesfw - Shell Script Framework tool
    - 200 SLOC, 20 functions
    - unified interface to kdialog, Xdialog, zenity
- shUnit2 - xUnit based unit testing for Unix shell scripts
- log4sh - logging framework
    - logging framework for shell scripts that works similar to http://logging.apache.org/
- bashworks - something messy
- rerun - a modular shell automation framework to organize your keeper scripts
    - 700 SLOC, 30 functions

Samples from books

- Learning the bash shell, 2005, samples, 62 files
- Bash Cookbook, 2007, 99 files
- Classic Shell Scripting, 2005. 82 files

Historical

- UNIX Power Tools, 1997, samples
- Portable Shell Programming, 1995, samples
    - 1000 SLOC, 33 files-functions
    - Regular operations implelented in shell. Usedfull on out of memory (OOM) when there is no memory to run external programms.
    - easy to use

References

- List of Bash shell-scripting libraries

See also

- Android Scripting

Developer Resources

Resources about linux kernel and its subsystems, including online documentation, books, podcasts and blabla.

> From: eLinux.org

Linux Kernel Resources

This page has references to various kernel resources (web sites and mailing lists) for developers. Most of this information was gathered over a year ago, and may not be accurate.

/\ Note: You should always look at the kernel MAINTAINERS file for up-to-date information

Contents

Vanilla Linux kernel

- www.kernel.org
- Linux Kernel Source Tarballs
- Linus' Git Repository

Mailing List (lkml)

- The Big List of Linux Kernel mailing lists, and where to find their archives
  - LKML - The Linux Kernel Mailing List (where the big boys hang out)
  - linux-embedded
    - Embedded Linux Kernel List
- How to subscribe to these lists

LKML summaries

- LWN Kernel page - Linux Weekly News kernel coverage

Repository access

- Kernel Git repositories
- Vanilla Linux Git Tree
  - This is "upstream". Get your code into here, please.
  - But this one has all the going back to 0.0.1, and updates itself from Linus's tree when you do a "git pull". (This is really cool. You want this.)

News

- Linux Weekly News, Kernel page

Changelog

- Comprehensible changelog of the linux kernel
    - http://wiki.kernelnewbies.org/LinuxChanges
- LWN atricles for spcific releases
    - 2.6.3 http://lwn.net/Articles/71669/
    - 2.6.10 http://lwn.net/Articles/117187/
    - 2.6.12 http://lwn.net/Articles/140165/
    - 2.6.15 http://lwn.net/Articles/166130/
- LWN aricles on 2.6 API changes
    - 2.6 API changes http://lwn.net/Articles/2.6-kernel-api/
    - 2.6.12 API changes http://lwn.net/Articles/140164/

Architecture Sites

MIPS

- web site = http://www.linux-mips.org/wiki/Main_Page
- mailing list = http://www.linux-mips.org/wiki/Net_Resources#Mailing_lists
- Maintainer = Ralph Baechle
- there's an alternate site on Source Forge
- the site is: http://sourceforge.net/projects/linux-mips
- Note that this is used for experimental stuff that hasn't been merged into the official mips tree by Ralph Baechle

ARM

- web site = http://www.arm.linux.org.uk/
- cvs access = http://cvs.arm.linux.org.uk/
- mailing list = http://www.arm.linux.org.uk/armlinux/mailinglists.php
- wiki = http://www.linux-arm.org/
- Maintainer = Russell King

PowerPC

- web site = http://penguinppc.org/
- mailing lists = http://penguinppc.org/about/community.php#lists
- Git repository = kernel.org:/pub/scm/linux/kernel/git/paulus/powerpc.git
- Maintainer = Paul Mackerras
- Power Macintosh Maintainer = Benjamin Herrenschmidt
- cross-compiler mini-howto: http://penguinppc.org/embedded/cross-compiling/

See the following for information on different linuxppc source trees available: http://www.penguinppc.org/dev/kernel.shtml

SuperH (SH)

- www.linux-sh.org
- oss.renesas.com/

- Git repository = kernel.org:/pub/scm/linux/kernel/git/lethal/sh-2.6.git
- mailing list address = linux-sh@vger.kernel.org
- mailing list page = http://vger.kernel.org/vger-lists.html#linux-sh
- mailing list archives = http://news.gmane.org/gmane.linux.ports.sh.devel
- wiki = http://linux-sh.org/shwiki/FrontPage
- Maintainer = Paul Mundt

Documentation

Online

- Rusty Russell's Unreliable Guide to Locking
- Embedded Linux kernel and driver development - Free Tutorials at Free Electrons
- Linux USB drivers - USB Driver Tutorial at Free Electrons

Books

- *Linux Kernel Developmen*t by Robert Love
  - Good introduction to Linux kernel development
- *Linux Device Drivers* by Jonathan Corbet, Alessandro Rubini, and Greg Kroah-Hartman
  - Great book for getting started with Linux device drivers
  - Free online pdf edition: http://lwn.net/images/pdf/LDD3/
  - online html http://www.makelinux.net/ldd3/
- *Essential Linux Device Drivers* by Sreekrishnan Venkateswaran
  - Introduction to driver development for major subsystems
- *Professional Linux Kernel Architecture* by Wolfgang Mauerer
  - Introduction to the architecture, concepts and algorithms of the Linux kernel
- *Understanding the Linux Kernel* by Daniel Bovet and Marco Cesati
  - Guided tour of the code that forms the core of all Linux operating systems
- *Linux Kernel in a Nutshell* by Greg Kroah-Hartman
  - Overview of kernel configuration and building
  - Free online edition: http://www.kroah.com/lkn/

Cross-reference / code online

- http://www.makelinux.net/kernel_map
- http://lxr.free-electrons.com/
- http://sosdg.org/~coywolf/lxr/source/
- http://lxr.linux.no/source/
- http://lxr.devzen.net/source/
- Find a kernel function line

Category:

- Development Tools

> From: eLinux.org

Kernel Subsystems

- The TTY Demystified - excellent explanation of kernel tty system
- Device Tree - a structure used to describe system hardware at startup - can be passed or modified by firmware, or built into kernel

> From: eLinux.org

Online Documentation

- Papers from the Ottawa Linux Symposium
- Free Software tools for embedded systems
- Real time in embedded Linux systems
- Embedded Linux optimizations
- Audio in embedded Linux systems
- Multimedia in embedded Linux systems
- Embedded Linux From Scratch... in 40 minutes!
- Linux technology reference

> From: eLinux.org

Category:Books

There are many books on Embedded Linux. This page lists several that received their own page on this wiki.

- Embedded Linux System Design and Development
    - by P. Raghavan, Amol Lad, and Sriram Neelakandan (Hardcover - Dec 21, 2005)
    - This one looks quite good.
- Embedded Linux Primer: A Practical Real-World Approach - by Christopher Hallinan
    - So does this one.
- Building Embedded Linux Systems
    - by Karim Yaghmour
    - Be sure to get the second edition (from 2008). The first edition (from 2003) is outdated.
- Advanced Programming in the UNIX Environment, Second Edition by the late W. Richard Stevens and Stephen A. Rago
    - Not embedded specific, but THE reference for Linux/Unix programming
- Linux System Programming
    - by Robert Love
    - Good introduction to Linux system programming
- PCI Express System Architecture
    - Mindshare Inc. (Author), Ravi Budruk (Author), Don Anderson (Author), Tom Shanley (Author)
- Linux Debugging and Performance Tuning
    - by Steve Best
- OMAP and DaVinci Software for Dummies
    - by Steve Blonstein, Alan Campbell, Texas Instruments
- Essential Linux Device Drivers
- Linux Device Drivers
- The_Linux_Programming_Interface_-_by_Michael_Kerrisk
    - The Linux Programming Interface by Michael Kerrisk

Pages in category "Books"

The following 11 pages are in this category, out of 11 total.

B

- Building Embedded Linux Systems

E

- Embedded Linux Primer
- Embedded Linux System Design and Development
- Essential Linux Device Drivers

F

- Flameman/books

L

- Linux Debugging and Performance Tuning
- Linux Device Drivers
- Linux Kernel Development - by Robert Love

O

- OMAP and DaVinci Software for Dummies

T

- The Art of Debugging with GDB, DDD, and Eclipse
- The Linux Programming Interface - by Michael Kerrisk

Category:

- Categories

> From: eLinux.org

Reference Material

- ARM Processor Reference Manuals - Registration required, but it's free.
  - go to http://infocenter.arm.com/ => ARM architecture => Reference Manuals => ... => registration link (only name, e-mail address and company name are strictly required).
- The UHAPI Forum standardizes hardware-independent application programming interfaces (APIs) for analog and digital televisions, set top boxes, DVD players and recorders, personal video recorders (PVRs), home servers and other consumer audio/video (A/V) devices.
- Hello_World_in_C
- How traceroute works!
- How DNS works!
- Order_One_Study
- RTWG-discussion-points
- Subset_Libc_Specification
- Extern_Vs_Static_Inline
- Slab_allocator
- Size_Tunables

> From: eLinux.org

Podcasts

- [1] - The (Original) Linux Link Tech Show, weekly Linux podcast with archive going back to 2003.
- [2] - Timesys LinuxLink Radio. (Despite the name, it's has nothing to do with the older Linux Link podcast, and it's not on the radio. No longer updates on a regular schedule, but the archives are available.)

> From: eLinux.org

Beginning Programming

This page is intended to help beginners get started with learning programming. Eventually, I'd like to provide a whole series of steps, exercises and tutorials about programming, to help anyone who would like to get involved with software development or game development.

Contents

Programming toolkits

- scratch - This is a beginning programming toolkit produced by MIT. This has numerous examples and tutorials, and is highly recommended as an excellent starting place for beginning programmers.
    - Free download available from: http://scratch.mit.edu/
    - http://www.youtube.com/watch?v=pkhjX792yVI
        - a good starter tutorial on scratch
    - Scratch Getting Started Guide
    - SNAP is scratch implemented in javascript - http://snap.berkeley.edu/
- Kahn Academy - Excellent online introductory course for computer science and programming, using interactive javascript
    - Computer Science tutorials and videos - Start by clicking on the Introductory Video.
    - Note that the bottom area of the screen is a tabbed dialog with different areas:

"Questions Tips & Feedback Spin-Offs Documentation". "Spin-Offs" has sample programs related to the tutorial, video or program you're looking at, and "Documentation" has description of different functions or statements you can use in your programs.

Online resources

- http://www.codeacademy.com/ - this site has good starting tutorials for javascript, html, php, python and ruby.
- http://www.w3schools.com/

Programming checklist

Follow these steps to learn how to program:

The very basics

- learn what a program statement is (and what a "computer language" is)
- learn what a variable is
- learn what a conditional (if statement) is
- learn what a loop is
- learn what input is
  - different kinds of input (key press, mouse move, mouse button)
- learn what output is
  - different kinds of output (text, image, sound)

Simple program 1: Write a program that counts how many times someone presses the space bar.

Simple program 2: Write a program that counts how many times someone presses each arrow key.

program 3: Write a program that starts counting when someone presses the space bar, and stops counting when they hit it a second time (like a stop-watch).

Program 4: Write a program that moves a ball to the right on the screen, when the right arrow is pressed.

Program 5: Write a program that starts a ball moving when key is pressed. The ball should keep moving until it hits a wall.

Note: For any of these programs, you can start with the program you used previously, and expand or modify it to do the next task. You should save it under a new name.

Program 6: Write a program that starts a ball moving in a different direction, depending on which arrow key (up, down, left, right) is pressed.

Program 7: Write a program that starts with a moving ball. When the ball hits a wall, it moves in the opposite direction.

Program 8: Combine programs 6 and 7. Write a program that starts a ball in a direction based on the key pressed. The ball should keep moving after the key is released. It should bounce off the wall and go the other direction when it hits a wall.

Languages

Programming can be done in many different computer languages.

Web Programming

Programming can be done in a web browser. This is a computer program, like Internet Explorer, Firefox or Chrome, that reads web pages from the Internet and shows them on your screen. Embedded devices, like phones, tablets and TVs also have web browsers.

In order to do web programming, you first need to learn how the browser presents information from a web page on your computer screen. It does this by processing the words on the web page (called "parsing" the page), and then drawing things like text, lines and images on your computer screen (usually, inside the browser window).

HTML

The words on a web page are part of a language called HTML (HyperText Markup Language). You can learn about this language here: http://www.w3schools.com/html/default.asp

Basically, the words and symbols are put into a file (or returned from a program running on the web server), and these words tell the web browser what to draw on the screen. The process of drawing things on the screen is called "rendering" the page.

Scratch

Here is a tutorial for a bouncing ball demo in scratch: http://scratch-time.blogspot.com/2008/12/how-to-make-bouncing-ball.html

Javascript

Here is a tutorial for a bouncing ball demo in Javascript: http://sixrevisions.com/html/bouncing-a-ball-around-with-html5-and-javascript/

Before you understand this language, and how it works, you first have to understand HTML.

Javascript resources

- http://www.webmonkey.com/2010/02/javascript_tutorial/
  - 5-part introductory tutorial
- http://www.w3schools.com/js/ - nice series of exercises on javascript

Python (with PyGame)

Here is an introduction to pygame, which has an bouncing ball demo as part of the intro.

http://www.pygame.org/docs/tut/intro/intro.html

C

One of the primitive programming languages it was developed by Dennis Ritchie, it has set the base for the languages that we see now. It is extensively used in system programming (Operating Systems, Device Drivers etc.). C is a powerful language with a primitive syntax, it is a compiled language unlike python. It is the lingua franca for a system developer. The linux kernel is written in 'C'.

Book: The C Programming language by Brian Kernighan and Dennis Ritchie

Online Resources: http://www.learn-c.org/ (Interactive)

Tips and Tricks

Tips and Tricks about chip identification, coding styling, debugging and GDB/GCC.

> From: eLinux.org

Chip Identification

Chip Detective: TV show helps with tech issue

One of the common problems while debugging a new board, as well as reverse engineering designs, is the inability to read part numbers on IC packages. During assembly the parts are baked at high temperatures and often washed with chemicals which can cause the package markings to become very difficult to read.

About 8 years ago i was watching one of the generic "Detective" TV shows where they had a hand gun with the serial number ground off. On the TV show, they used a colored die and different camera lighting to bring out the serial number as the imprint what still on the metal of the gun. I thought to myself... Can i use this method to read the markings on IC packages?

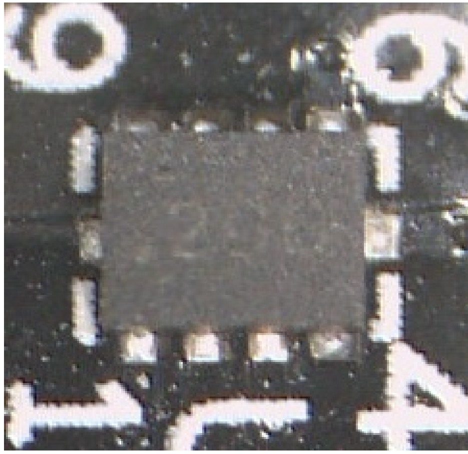
I purchased a small usb based microscope and did some testing. Sure enough it worked.

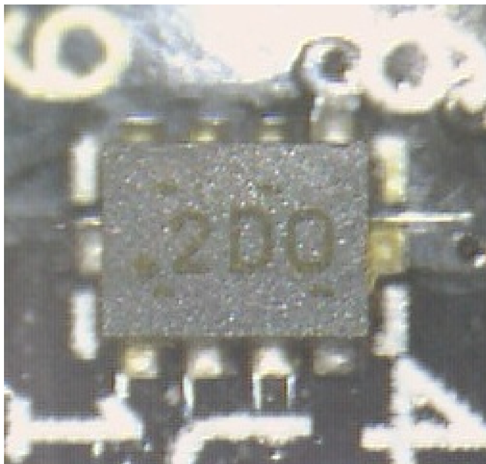
Today one of my most valued tools is the Celestron 44302-A Deluxe Handheld Digital Microscope(about $50USD). It works with most linux utilities including guvcview which I use to capture images. I've uploaded a picture of an IC taken after the IC had been installed on a pcb. You will notice that the markings are pretty hard to read. Using a Sharpie you can paint the top of the package to bring out the markings. I've tried all the basic colors and found that the yellow is the best. It is dark enough to bring out the markings, but not too opaque to obscure the markings. The second photo is a capture after painting the IC with the yellow Sharpie. You can see a pretty dramatic difference. The third capture is done with the usb microscope at a 45 degree angle with the part, which makes the markings really jump out.

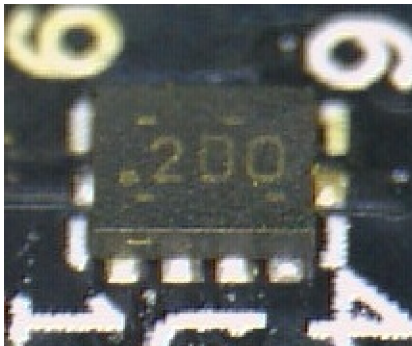
EDIT: this also works on ceramic packages where the markings have intentionally removed. Since most of the markings are done with laser etching, the ceramics usually retain the markings and the yellow ink will get absorbed into the ceramic portions "damaged" by the laser etching.

- Celestron Microscope






2D0


2D0


Sharpie

> From: eLinux.org

Code Styling Tips

Here are some miscellaneous tips for good code styling:

Proper Linux Kernel Coding Style

See the kernel coding style guide in any kernel source tree at: Documentation/CodingStyle (Online here)

Greg Kroah-Hartman wrote some additional tips in his article: Proper Linux Kernel Coding Style

Michael S. Tsirkin made a kernel guide to space (*a boring list of rules*) which got polished on a worth reading thread in LKML in 2005.

use of #ifdefs

Rob Landley writes:

Read: http://doc.cat-v.org/henry_spencer/ifdef_considered_harmful.pdf

Personally, I tend to have symbols #defined to a constant 0 or 1 depending on whether or not a function is enabled, and then just use if(SYMBOL) as a guard and let the compiler's dead code eliminator take it out for me at compile time (because if(0) {blah;} shouldn't put any code in the resulting .o file with any optimizer worth its salt. Borland C for DOS managed simple dead code elimination 20 years ago...)

See also

Sparse

Category:

- Development Tools

> From: eLinux.org

Debugging Tips

- See the Debugging_Portal page
- See the Kernel Debugging Tips page
- See also Debugging Makefiles
- Debugging by printing
- Debug user-space initialization:
  - If you get a panic - "not syncing: Attempted to kill init!" it can be for many different reasons. Try setting CONFIG_DEBUG_USER=y in your .config and pass 'user_debug=255' in the kernel command line. That will give you a more verbose output about why user space programs crash. Thanks to Daniel Mack on the linux-arm-kernel mailing list for this tip.

Category:

- Tips and Tricks

> From: eLinux.org

GDB Tips

get element size

Sometimes, with complex structures (arrays of structs containing nested structs or arrays), it is hard to determine the actual size of a particular element.

You can use gdb with a program image to get the size of structures, by looking at the offset of an element of the structure relative to an address of zero:

Here are some examples:

```
  ${CROSS_COMPILE}gdb vmlinux
GNU gdb (GDB) 7.2
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-pc-linux-gnu --target=arm-sony-linux-gnueabi".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
(gdb) p &((struct poll_wqueues *)0)->polling_task
$6 = (struct task_struct **) 0xc
(gdb) p/d &((struct poll_wqueues *)0)->error
$4 = 20
```

the second example could be read as: "print, in decimal, the address (offset) of the element error using address 0 cast as a pointer to struct poll_wqueues"

'pt' is the first element of struct poll_wqueues. Here is an example using array offsets, showing that struct poll_wqueues is 604 bytes long.

```
(gdb) p/d &((struct poll_wqueues *)0)[0]->pt
$2 = 0
(gdb) p/d &((struct poll_wqueues *)0)[1]->pt
$3 = 604
```

Category:

- Tips and Tricks

> From: eLinux.org

GCC Tips

Contents

What's Here, Why You Should Care

A collection of tips useful to those doing embedded development. Accumulated over several years of doing project work, helping other engineers, untangling projects for customers and feedback from several CELF presentations related to this topic.

Tips

View Compilation Plan

```
gcc -### <the rest of your command line goes here>
```

The GCC you run is a driver program for a bunch of other programs. With this parameter, gcc will produce (but not actually execute) the commands it would have used to accomplish the task you asked it to do. This way, you can see the gory details of what's going on behind the scenes. What library is being used? What is -mcpu set to? It's all there.

You can pipe this output to a file and execute that to compile a program, making it easy to experiment with tweaks to the linker or assembler.

```
Reading specs from /opt/timesys/toolchains/ppc7xx-linux/lib/gcc/powerpc-linux/3.4.1/specs
Configured with: ../configure --prefix=/opt/timesys/toolchains/ppc7xx-linux --mandir=/opt/timesys/toolchains/ppc7xx-l
inux/share/man --infodir=/opt/timesys/toolchains/ppc7xx-linux/share/info --enable-shared --enable-threads=posix --dis
able-checking --with-system-zlib --enable-__cxa_atexit --disable-libunwind-exceptions --enable-languages=c,c++ --with
-sysroot=/here/workdir/i386-x-ppc7xx/deleteme --disable-libgcj --build=i686-timesys-linux --host=i686-timesys-linux -
-target=powerpc-linux --program-prefix=ppc7xx-linux-
Thread model: posix
gcc version 3.4.1 20040714 (TimeSys 3.4.1-7)
 /opt/timesys/toolchains/ppc7xx-linux/libexec/gcc/powerpc-linux/3.4.1/cc1 -quiet -v -D__unix__ -D__gnu_linux__ -D__li
nux__ -Dunix -D__unix -Dlinux -D__linux -Asystem=linux -Asystem=unix -Asystem=posix -I/opt/timesys/toolchains/ppc7xx-
linux/powerpc-linux/include/nptl file.c -quiet -dumpbase file.c -auxbase file -version -o /tmp/ccShiHn4.s
ignoring nonexistent directory "/here/workdir/i386-x-ppc7xx/deleteme/usr/local/include"
ignoring nonexistent directory "/here/workdir/i386-x-ppc7xx/deleteme/usr/include"
#include "..." search starts here:
#include <...> search starts here:
 /opt/timesys/toolchains/ppc7xx-linux/powerpc-linux/include/nptl
 /opt/timesys/toolchains/ppc7xx-linux/lib/gcc/powerpc-linux/3.4.1/include
 /opt/timesys/toolchains/ppc7xx-linux/lib/gcc/powerpc-linux/3.4.1/../../../../powerpc-linux/include
End of search list.
GNU C version 3.4.1 20040714 (TimeSys 3.4.1-7) (powerpc-linux)
 compiled by GNU C version 3.2.2 20030222 (Red Hat Linux 3.2.2-5).
GGC heuristics: --param ggc-min-expand=47 --param ggc-min-heapsize=32138
 /opt/timesys/toolchains/ppc7xx-linux/lib/gcc/powerpc-linux/3.4.1/../../../../powerpc-linux/bin/as -mppc -many -V -Qy
 -o /tmp/ccWeV3a3.o /tmp/ccShiHn4.s
GNU assembler version 2.15.90.0.3 (powerpc-linux) using BFD version 2.15.90.0.3 20040415
 /opt/timesys/toolchains/ppc7xx-linux/libexec/gcc/powerpc-linux/3.4.1/collect2 --eh-frame-hdr -V -Qy -L/opt/timesys/t
oolchains/ppc7xx-linux/powerpc-linux/lib/nptl --rpath-link /opt/timesys/toolchains/ppc7xx-linux/powerpc-linux/lib/tls
 -m elf32ppclinux -dynamic-linker /lib/ld.so.1 -o file /opt/timesys/toolchains/ppc7xx-linux/lib/gcc/powerpc-linux/3.4
.1/../../../../powerpc-linux/lib/crt1.o /opt/timesys/toolchains/ppc7xx-linux/lib/gcc/powerpc-linux/3.4.1/../../../../
powerpc-linux/lib/crti.o /opt/timesys/toolchains/ppc7xx-linux/lib/gcc/powerpc-linux/3.4.1/crtbegin.o -L/opt/timesys/t
oolchains/ppc7xx-linux/lib/gcc/powerpc-linux/3.4.1 -L/opt/timesys/toolchains/ppc7xx-linux/lib/gcc/powerpc-linux/3.4.1
/../../../../powerpc-linux/lib /tmp/ccWeV3a3.o -lgcc --as-needed -lgcc_s --no-as-needed -lc -lgcc --as-needed -lgcc_s
 --no-as-needed /opt/timesys/toolchains/ppc7xx-linux/lib/gcc/powerpc-linux/3.4.1/crtsavres.o /opt/timesys/toolchains/
ppc7xx-linux/lib/gcc/powerpc-linux/3.4.1/crtend.o /opt/timesys/toolchains/ppc7xx-linux/lib/gcc/powerpc-linux/3.4.1/..
/../../../powerpc-linux/lib/crtn.o
GNU ld version 2.15.90.0.3 20040415
  Supported emulations:
   elf32ppclinux
   elf32ppc
   elf32ppcsim
```

Note: on my (Tim Bird's) Ubuntu 12.04 system, using gcc 4.6.3, it shows quite different output. Specifically, it is missing anything about the include directories.

See default include search paths

You can get a report about the include search paths with the following command:

```
$ gcc -xc -E -v -
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/4.6/lto-wrapper
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu/Linaro 4.6.3-1ubuntu5' --with-bugurl=file:///usr/share
/doc/gcc-4.6/README.Bugs --enable-languages=c,c++,fortran,objc,obj-c++ --prefix=/usr --program-suffix=-4.6 --enable-s
hared --enable-linker-build-id --with-system-zlib --libexecdir=/usr/lib --without-included-gettext --enable-threads=p
osix --with-gxx-include-dir=/usr/include/c++/4.6 --libdir=/usr/lib --enable-nls --with-sysroot=/ --enable-clocale=gnu
 --enable-libstdcxx-debug --enable-libstdcxx-time=yes --enable-gnu-unique-object --enable-plugin --enable-objc-gc --d
isable-werror --with-arch-32=i686 --with-tune=generic --enable-checking=release --build=x86_64-linux-gnu --host=x86_6
4-linux-gnu --target=x86_64-linux-gnu
Thread model: posix
gcc version 4.6.3 (Ubuntu/Linaro 4.6.3-1ubuntu5)
COLLECT_GCC_OPTIONS='-E' '-v' '-mtune=generic' '-march=x86-64'
 /usr/lib/gcc/x86_64-linux-gnu/4.6/cc1 -E -quiet -v -imultilib . -imultiarch x86_64-linux-gnu - -mtune=generic -march
=x86-64 -fstack-protector
ignoring nonexistent directory "/usr/local/include/x86_64-linux-gnu"
ignoring nonexistent directory "/usr/lib/gcc/x86_64-linux-gnu/4.6/../../../../x86_64-linux-gnu/include"
#include "..." search starts here:
#include <...> search starts here:
 /usr/lib/gcc/x86_64-linux-gnu/4.6/include
 /usr/local/include
 /usr/lib/gcc/x86_64-linux-gnu/4.6/include-fixed
 /usr/include/x86_64-linux-gnu
 /usr/include
End of search list.
```

Timing sub-commands

You can get a report for the amount of time taken for each sub-command that gcc uses, using the '-time' command.

Example:

```
$ gcc -time -static hello.c -o hello
# cc1 0.01 0.01
# as 0.00 0.00
# collect2 0.03 0.02
$
```

This was on a fast machine, with a very small sample program. It shows that the sub-programs 'cc1', 'as', and 'collect2' (the compiler, the assembler, and the linker (collect2 is a front-end for 'ld'), respectively), took less than a few hundredths of a second each to run.

Pre-Process, Retain Comments

```
gcc -C -E <file-name.c> -o file
```

Some engineers love to do coding in macros. The rest of us would like to break their fingers. This command will run the file through the pre-processor, expanding all macros, but retaining all comments. Stick a comment like "LOOK HERE" and search for that so you reduce the amount of time you spend looking for the offending code.

Pre-Process, Retain Defines/Macros

```
gcc -dD -E <file-name.c> -o file
```

Just like the previous example, but perhaps you're trying to trace the macro define mess.

See what Files the Linker is Using

```
gcc -Wl,-t <parameters>
```

Displays what files the linker opens in what order. When looking in archive files, the archive file is displayed in para theses, followed by the file in the archive. Very handy when working through a legacy project that depends on files linking in a certain order that suddenly breaks because of a small (probably viewed as not noteworthy) change in a makefile somewhere.

```
/usr/bin/ld: mode elf_i386
/usr/lib/gcc-lib/i386-redhat-linux/3.3.3/../../../crt1.o
/usr/lib/gcc-lib/i386-redhat-linux/3.3.3/../../../crti.o
/usr/lib/gcc-lib/i386-redhat-linux/3.3.3/crtbegin.o
/tmp/cc37FxnS.o
-lgcc_s (http://eLinux.org/usr/lib/gcc-lib/i386-redhat-linux/3.3.3/libgcc_s.so)
/lib/libc.so.6
(http://eLinux.org/usr/lib/libc_nonshared.a)elf-init.oS
-lgcc_s (http://eLinux.org/usr/lib/gcc-lib/i386-redhat-linux/3.3.3/libgcc_s.so)
/usr/lib/gcc-lib/i386-redhat-linux/3.3.3/crtend.o
/usr/lib/gcc-lib/i386-redhat-linux/3.3.3/../../../crtn.o
```

Print Pre-defined Macros

```
gcc -E -dM - < /dev/null | cut -c 9- | sort
```

Very handy when porting code. Lets you know if your target processor has some missing defines or if something is different (like __INT_MAX__) that can have interesting effects on your project. Diff the output from the old to the new compiler so you can easily see the differences, makes it easy to spot problems before getting started.

Sample output, from a compiler targeting an ARM processor.

```
__APCS_32__ 1
__arm__ 1
__ARM_ARCH_4T__ 1
__ARMEL__ 1
__CHAR_BIT__ 8
__CHAR_UNSIGNED__ 1
__DBL_DENORM_MIN__ 4.9406564584124654e-324
__DBL_DIG__ 15
__DBL_EPSILON__ 2.2204460492503131e-16
__DBL_HAS_DENORM__ 1
__DBL_HAS_INFINITY__ 1
__DBL_HAS_QUIET_NAN__ 1
__DBL_MANT_DIG__ 53
__DBL_MAX_10_EXP__ 308
__DBL_MAX__ 1.7976931348623157e+308
__DBL_MAX_EXP__ 1024
```

Mixed Assembler and Source Output

```
gcc -g somefile.c -o somefile
objdump -S somefile
```

Prints out each line in the program and the corresponding assembly code. Very handy when you're trying to see that the processor is generating the correct code, with the instructions you're expecting. You can also see the effects of optimization, but would recommend doing this for a small amount of code because when the optimization level is high, there's a much lower relationship between line of code and generated assembler.

Here's an example of what objdump produces for a few lines of code:

```
  gpvSharedMemory = shmat(hSharedMemory, NULL, 0);
10000958:       80 7f 00 10     lwz     r3,16(r31)
1000095c:       38 80 00 00     li      r4,0
10000960:       38 a0 00 00     li      r5,0
10000964:       48 01 09 31     bl      10011294 <shmat@plt>
10000968:       7c 60 1b 78     mr      r0,r3
1000096c:       3d 20 10 01     lis     r9,4097
10000970:       90 09 11 d0     stw     r0,4560(r9)
  if (errno != 0) {
10000974:       48 01 08 d1     bl      10011244 <__errno_location@plt>
10000978:       7c 60 1b 78     mr      r0,r3
1000097c:       7c 09 03 78     mr      r9,r0
10000980:       80 09 00 00     lwz     r0,0(r9)
10000984:       2f 80 00 00     cmpwi   cr7,r0,0
10000988:       41 9e 00 50     beq-    cr7,100009d8 <main+0x10c>
```

Specify Language

```
gcc -x c a-c-source-file.with-a-non-standard-extension -o test.out
```

Great for legacy projects where where the file extensions don't match with GCC's expectations, while less of a problem since many projects got their start with GCC, this still is an issue with long-running projects that years back, used some other compiler. This stays in effect for the following file on the command line.

List Include File Dependencies

There's a whole family of things around -M. These produce a rule that could be used in a make file, with the included files as dependencies.

```
gcc -M <file name>
```

This shows you all includes, even those on the system path. Useful if you're doing porting work or validating if your compiler is working as expected and getting the files from the right place. You'll see something like this for a basic hello world program

```
hello.o: hello.c /usr/include/stdio.h /usr/include/features.h \
  /usr/include/sys/cdefs.h /usr/include/gnu/stubs.h \
  /usr/lib/gcc-lib/i386-redhat-linux/3.3.3/include/stddef.h \
  /usr/include/bits/types.h /usr/include/bits/wordsize.h \
  /usr/include/bits/typesizes.h /usr/include/libio.h \
  /usr/include/_G_config.h /usr/include/wchar.h /usr/include/bits/wchar.h \
  /usr/include/gconv.h \
  /usr/lib/gcc-lib/i386-redhat-linux/3.3.3/include/stdarg.h \
  /usr/include/bits/stdio_lim.h /usr/include/bits/sys_errlist.h

gcc -MM <file name>
```

Like -M, but no system files. Great to see if your project is configured and working as expected.

```
gcc -M -MG <file name>
```

The prior -M and -MM commands will stop if header files can be located. The parameter -MG will just produce the dependency list with the missing file. Engineers that have projects that generate header files as part of the build find -MM very handy.

```
gcc -M -MT '<target>' <file name>
```

By default, the target will be the \.o This command will make the default the value of \.

If the command was

```
gcc -M -MT '$(target)' hello.c
```

You would see

```
$(target): hello.c
```

Symbol Trace

```
gcc -Wl,-y,printf hello.c
```

This is very handy when you want to understand where the linker is finding a definition of a symbol. Some projects have name collisions or link order dependencies. This lets you see precisely what the linker is doing.

Given a hello world program, you would see output like

```
/tmp/ccwZx5UV.o: reference to printf
/lib/libc.so.6: definition of printf
```

The reference is in a temporary file created during the compilation process. If you were linking several object files together explicitly, you would see the name of the object file where printf was referenced.

Saving temporary files

```
gcc -save-temps hello.c
```

The compilers temporary files are saved. This is often invaluable dealing with complicated makefiles to peek at the preprocessed output without having to figure out how to do a -E option by hand.

Categories:

- Development Tools
- Tips and Tricks
- Compiler

Misc & Wishlist

About bluetooth network, continuous logging and crash diagnostics...

> From: eLinux.org

Bluetooth Network

This page has information about setting up a Bluetooth Personal Area Network (PAN) with BlueZ. Having a Bluetooth network is helpful for providing network access to low power embedded devices.

Contents

Background

Information on piconets can be found on Wikipedia Basic information on BlueZ PAN support can be found here: [1]

Limitations

A PAN network is limited to 7 clients and provides substantially less bandwidth (~700Kbit/s) than other WiFi networks.

Requirements

To setup a home piconet, you'll need:

- A Bluetooth device, such as a USB dongle, preferably Class 1 for range purposes.

- A kernel that supports the Bluez stack including BNEP.
- bluez-utils (testing with 3.36).
- Kernel ethernet bridging support.
- bridge-utils (tested with 1.4).

These instructions are based on a Debian/Sid system, but the setup should be similar for other distributions.

BlueZ Configuration

Setup /etc/bluetooth/hcid.conf

hcid.conf(5) Your piconet server should advertise itself appropriately. Modify the class parameter within the device section so that the host presents itself as a network access point device offering network service:

```
# Local device class
class 0x020300;
```

Change your piconet server to prefer master role on incoming connections:

```
lm master;
```

Make your piconet server permanently discoverable:

```
discovto 0;
```

Daemon Configuration

pand(1) Setup the command line options for the pand daemon. Within Debian, this is done through the file /etc/default/bluetooth. The command lines for the pand daemon should be:

```
--listen --role NAP -u /etc/bluetooth/pan/dev-up -o /etc/bluetooth/pan/dev-down
```

End Result (When host is connected)

ifconfig bnep0

```
bnep0     Link encap:Ethernet  HWaddr 00:11:f6:05:79:95
          inet6 addr: fe80::211:f6ff:fe05:7995/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:23661 errors:0 dropped:0 overruns:0 frame:0
          TX packets:29381 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2976646 (2.8 MiB)  TX bytes:27249215 (25.9 MiB)
```

hcitool con

```
Connections:
    > ACL 00:1B:DC:0F:A8:AE handle 8 state 1 lm MASTER
```

Bridge Configuration

The kernel only provides an Ethernet device when at least one PAN client has connected. This means that there will be no associated device when no devices are connected. This can be very inconvenient when providing services such as DHCP. By utilizing Ethernet Bridging, a permanent pan0 device can be created.

Setup /etc/network/interfaces

interfaces(5)

bridge-utils-interfaces(5) On Debian systems, network interfaces are configured through this file. An example configuration would be:

```
auto pan0
iface pan0 inet static
        address 10.1.0.1
        netmask 255.255.255.0
        broadcast 10.1.0.255
        bridge_ports none
        bridge_fd 0
        bridge_stp off
```

Alternatively, the pan0 interface can be configured manually:

```
brctl addbr pan0
brctl setfd pan0 0
brctl stp pan0 off
ifconfig pan0 10.1.0.1 netmask 255.255.255.0
```

Setup /etc/bluetooth/pan/dev-{up|down}

The dev up/down files add and remove the bnep0 device from the pan0 bridge interface as the first device enters the network, and as the last device leaves the network.

/etc/bluetooth/pan/dev-up

```
#!/bin/sh
ifconfig $1 up
brctl addif pan0 $1
```

/etc/bluetooth/pan/dev-down

```
#!/bin/sh
brctl delif pan0 $1
ifconfig $1 down
```

End Result (When host is connected)

brctl show

```
bridge name bridge id       STP enabled interfaces
pan0        8000.0011f6057995   no      bnep0
```

ifconfig pan0

```
pan0      Link encap:Ethernet  HWaddr 00:11:f6:05:79:95
          inet addr:10.1.0.1  Bcast:10.1.0.255  Mask:255.255.255.0
          inet6 addr: fe80::200:ff:fe00:0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:30706 errors:0 dropped:0 overruns:0 frame:0
          TX packets:40037 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:3681538 (3.5 MiB)  TX bytes:34573855 (32.9 MiB)
```

Setup DHCP

Unless avahi zeroconf will be used to assign address, a DHCP server will be required.

Setup /etc/dhcpd.conf

Basic configuration:

```
option domain-name-servers <dns1>,<dns2>,<dns3>;

default-lease-time 864000;
max-lease-time 864000;

subnet 10.1.0.0 netmask 255.255.255.0 {
  option domain-name "blue";
  range 10.1.0.100 10.1.0.200;
  option routers 10.1.0.1;
}
```

Daemon Configuration

Setup the command line options for the dhcpd daemon. Within Debian, this is done through the file /etc/default/dhcp. Tho command lines for the dhcpd daemon should be:

```
pan0
```

Setup Network

If your piconet server is not the machine you intend to access your piconet devices from and/or your piconet devices need to access hosts other than your piconet server, routing and/or NAT will need to be configured

Shorewall

Adding your piconet to an existing Shorewall configuration is by far the easiest method.

params

```
BLUE_IF=pan0
```

interfaces

```
#ZONE   INTERFACE       BROADCAST       OPTIONS
blue    $BLUE_IF    detect      tcpflags,dhcp,detectnets,nosmurfs
```

zones

```
#ZONE   TYPE        OPTIONS     IN          OUT
#                   OPTIONS         OPTIONS
blue    ipv4
```

policy

Allow piconet to access Internet:

```
#SOURCE         DEST            POLICY          LOG             LIMIT:BURST
#                                               LEVEL
blue            net             ACCEPT
```

A rule like the following would allow the local network to access the piconet:

```
#SOURCE         DEST            POLICY          LOG             LIMIT:BURST
#
loc             all             ACCEPT
```

The last line of the policy file should of course contain an all/all DROP rule.

masq

Allow local network to access piconet masquerading as piconet server:

```
#INTERFACE              SUBNET          ADDRESS         PROTO   PORT(S) IPSEC
$BLUE_IF        $LOC_IF
```

Masquerade piconet network access to Internet

```
#INTERFACE              SUBNET          ADDRESS         PROTO   PORT(S) IPSEC
$NET_IF         $BLUE_IF
```

rules

Not allowing your open piconet to do things like Spam and/or access your Cable modem is probably a good thing.

```
#ACTION         SOURCE  DEST            PROTO   DEST    SOURCE          ORIGINAL        RATE            USER/
#                                               PORT    PORT(S)         DEST            LIMIT           GROUP
SMTP/REJECT     blue    net
DROP   blue    net:10.0.0.0/8,192.168.0.0/16,172.16.0.0/12
```

Netfilter

A very basic Netfilter setup, assuming that eth1 connects to the Internet, and eth0 connects to the local network.

```
# Enable masquerading access to the Internet (rule may already exists)
iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
# Enable masquerading access to the piconet from the local net
iptables -t nat -A POSTROUTING -i eth0 -o pan0 -j MASQUERADE
# Enable routing (may already exist)
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Network Manager 0.7

Network Manager provides connection sharing functionality. From the "Edit Connections" dialog, select "Add". Name the connection bnep0 and enter the Bluetooth device's MAC address into the Wired tab. Select "Shared to other computers" on the "IPv4 Settings" tab.

Client Device

Embedded devices should execute the command:

```
pand --connect <bdaddr of piconet server> --persist -u ifup -o ifdown
```

Upon boot, alternatively, the following command can be used:

```
pand --search --persist -u ifup -o ifdown
```

/etc/network/interfaces file

This step applies to Debian and Debian like (Angstrom/OE) distributions. Modification will be required for other distributions:

```
# Bluetooth networking
allow-hotplug bnep0
iface bnep0 inet dhcp
```

End result

ifconfig bnep0

```
bnep0     Link encap:Ethernet  HWaddr 00:1B:DC:0F:A8:AE
          inet addr:10.1.0.100  Bcast:10.1.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:29272 errors:0 dropped:0 overruns:0 frame:0
          TX packets:23598 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:27242050 (25.9 MiB)  TX bytes:2964918 (2.8 MiB)
```

route -n

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
10.1.0.0        0.0.0.0         255.255.255.0   U     0      0        0 bnep0
0.0.0.0         10.1.0.1        0.0.0.0         UG    0      0        0 bnep0
```

hcitool con

```
Connections:
    < ACL 00:11:F6:05:79:95 handle 42 state 1 lm SLAVE
```

Category:

- Networking

> From: eLinux.org

Continuous Logging for Watchdog Timer Expiration

Implementation Idea

I think this can be achieved by modifying the driver that services the hardware watchdog. This driver is aware of the hardware timeout (say this is N) and the application can make it a rule to always service the hardware watchdog device (http://eLinux.org/dev/wtd) every N/2 seconds. The driver can now warn when it has not been serviced during the first N/2 and has N/2 left to dump this event somehow.

--Leon Woestenberg

Category:

- Kernel

> From: eLinux.org

Crash Diagnostics

kernel crash debugging technique using "crash" utility.

Introduction

Most frequent issue in debugging kernel programs is lack of information; lack of information in terms of stack trace, kernel logs and crash screen-shots. Crash utility provides solution for this. It provides gdb like interface for debugging.

Prerequisites

```
vmlinux object file.
```

This is kernel image with debugging information. By default, we get compressed kernel image with no debug information. For getting, vmlinux object file, you need to install "kernel-debuginfo" rpm. If you want to debug any module, make sure it is compiled with gcc flag '-g'.

Installation

We receive crash utility as rpm.

```
rpm name is of format: crash-$version.$arch.rpm
```

Here, $version shows version number and $arch shows system architecture.

For getting crash dump: For getting crash dump, we need to add "crashkernel" option to grub command line. i.e. we can have:

```
ro root=LABEL=/1 rhgb quiet crashkernel=128M@16M
```

instead of,

```
ro root=LABEL=/1 rhgb quiet
```

Here, parameter "crashkernel=128M@16M" reserves 128MB of physical memory starting at 16MB. This reserved memory is used to preload and run the capture kernel (i.e. to capture crash dump). This command line option ensures that, whenever crash occurs, it stores crash dump at "/var/crash" and they are stored according to date and time.

There are other options to get crash dump like diskdump, netdump, etc.

Running crash Utility: 1. Debugging last kernel panic:

```
#crash <patht to vmlinux> /var/crash/crash_dump_name
```

1. Watching current running kernel:

   ```
   #crash
   ```

This will prompt to crash shell. Just as example, following are commands:

```
crash> help
crash> bt                    ---> for backtrace
crash> log                  ---> for dumping current system buffer
```

For more information on this, you can visit at: http://www.redhatmagazine.com/2007/08/15/a-quick-overview-of-linux-kernel-crash-dump-analysis/ http://people.redhat.com/anderson/crash_whitepaper/

Category:

- Development Tools

Debugging Portal

This chapter is the new portal for all debugging related stuff. It currently deals with Kernel topics, but feel free to add user space debugging too.

Kernel Debugging

About kernel debugging tools and tips.

> From: eLinux.org

Debugging by printing

Probably the simplest way to get some debug information from your kernel code is by printing out various information with the kernel's equivalent of printf - the printk function and its derivatives. The k in printk is used to specifically remind kernel developers that the environment is different.

Contents

Usage

printk works more or less the same way as printf in userspace, so if you ever debugged your userspace program using printf, you are ready to do the same with your kernel code, e.g. by adding:

```
printk("My Debugger is Printk\n");
```

This wasn't that difficult, was it?

Usually you would print out some more interesting information like

```
printk("Var1 %d var2 %d\n", var1, var2);
```

just like in userspace.

In order to see the kernel messages, just use the

```
$ dmesg
```

command in one of your shells - this one will print out the whole kernel log buffer to you.

Most of the conversion specifiers supported by the user-space library routine printf() are also available in the kernel; there are some notable additions, including "%pf", which will print the symbol name in place of the numeric pointer value, if available.

The supported format strings are quite extensively documented in Documentation/printk-formats.txt

However please **note**: always use *%zu*, *%zd* or *%zx* for printing *size_t* and *ssize_t* values. ssize_t and size_t are quite common values in the kernel, so please use the *%z* to avoid annoying compile warnings.

Author's practical advice: If you want to debug an oops (e.g caused by releasing a resource twice) in your driver and you don't have a clue where the oops happens, simply add this line

```
printk(KERN_ALERT "DEBUG: Passed %s %d \n",__FUNCTION__,__LINE__);
```

after each possibly offending statement. Recompile and (re-)load the module and trigger the error condition - your log now shows you the last line that was successfully executed before the oops happened.

Of course you should **remove** these 'rude' statements before shipping your module ;)

Log Levels

If you look into real kernel code you will always see something like:

```
printk(KERN_ERR "something went wrong, return code: %d\n",ret);
```

where "*KERN_ERR*" is one of the eight different log levels defined in include/linux/kern_levels.h and specifies the severity of the error message.

Note that there is **NO** comma between the *KERN_ERR* and the format string (as the preprocessor concatenates both strings)

The log levels are:

Name	String	Meaning	alias function
KERN_EMERG	"0"	Emergency messages, system is about to crash or is unstable	pr_emerg
KERN_ALERT	"1"	Something bad happened and action must be taken immediately	pr_alert
KERN_CRIT	"2"	A critical condition occurred like a serious hardware/software failure	pr_crit
KERN_ERR	"3"	An error condition, often used by drivers to indicate difficulties with the hardware	pr_err
KERN_WARNING	"4"	A warning, meaning nothing serious by itself but might indicate problems	pr_warning
KERN_NOTICE	"5"	Nothing serious, but notably nevertheless. Often used to report security events.	pr_notice
KERN_INFO	"6"	Informational message e.g. startup information at driver initialization	pr_info
KERN_DEBUG	"7"	Debug messages	pr_debug, pr_devel if DEBUG is defined
KERN_DEFAULT	"d"	The default kernel loglevel	
KERN_CONT	""	"continued" line of log printout (only done after a line that had no enclosing \n) [1]	pr_cont

Note that the actual values of the log levels are prepended by the KERN_SOH character whose ASCII value is '\001'. Read the source for more details.

The pr_* macros (with exception of pr_debug) are simple shorthand definitions in *include/linux/printk.h* for their respective printk call and should probably be used in newer drivers.

pr_devel and *pr_debug* are replaced with *printk(KERN_DEBUG ...* if the kernel was compiled with *DEBUG*, otherwise replaced with an empty statement.

For drivers the pr_debug should not be used anymore (use dev_dbg instead).

If you don't specify a log level in your message it defaults to *DEFAULT_MESSAGE_LOGLEVEL* (usually *"4"=KERN_WARNING*) which can be set via the *CONFIG_DEFAULT_MESSAGE_LOGLEVEL* kernel config option (*make menuconfig-> Kernel Hacking -> Default message log level*)

The log level is used by the kernel to determine the importance of a message and to decide whether it should be presented to the user immediately, by printing it to the current console (where console could also be a serial line or even a printer, not an xterm).

For this the kernel compares the log level of the message to the *console_loglevel* (a kernel variable) and if the priority is higher (i.e. a lower value) than the *console_loglevel* the message will be printed to the current console.

To determine your current *console_loglevel* you simply enter:

```
$ cat /proc/sys/kernel/printk
    7       4       1       7
    current default minimum boot-time-default
```

The first integer shows you your current console_loglevel; the second the default log level that you have seen above.

To change your current console_loglevel simply write to this file, so in order to get all messages printed to the console do a simple

```
# echo 8 > /proc/sys/kernel/printk
```

and every kernel message will appear on your console.

Another way to change the console log level is to use *dmesg* with the *-n* parameter

```
# #set console_loglevel to print KERN_WARNING (4) or more severe messages
# dmesg -n 5
```

Only messages with a value lower (**not** lower equal) than the console_loglevel will be printed.

You can also specify the console_loglevel at boot time using the *loglevel* boot parameter. (see Documentation/kernel-parameters.txt for more details)

Author's practical advice: Of course you should always specify an appropriate log level for your messages, but for **debugging**, I guess most developers leave the console_loglevel unchanged and simply use KERN_ERR or KERN_CRIT to ensure the message reaches the console.

```
pr_err("REMOVE ME: my debug statement that I swear to remove when I'm done\");
```

Please make sure to remove these 'inappropriately' tagged messages before shipping the module ;)

KERN_CONT and *pr_cont* are special cases since they do not specify an urgency but rather indicate a 'continued message' e.g.:

```
printk(KERN_ERR "Doing something was ");
/* <100 lines of whatever>*/
if (success)
   printk(KERN_CONT "successful\n");
else
   printk(KERN_CONT "NOT successful\n");

-> "Doing something was successful"
```

Important Note: *KERN_CONT* and *pr_cont* should only be used by core/arch code during early bootup (a continued line is not SMP-safe otherwise).[2]

Rate limiting and one time messages

Occasionally you have to insert a message in a section which gets called quite often. This not only might have a severe performance impact, it also could overwrite and spam your kernel buffer so it should be avoided.

As always the kernel already provides you with nice functions that solve your problems:

```
printk_ratelimited(...)
```

and

```
printk_once(...)
```

printk_once is fairly trivial - no matter how often you call it, it prints once and never again.

printk_ratelimited is a little bit more complicated - it prints by default not more than 10 times in every 5 seconds (for each function it is called in).

If you need other values for the maximum burst count and the timeout, you can always setup your own ratelimit using the *DEFINE_RATELIMIT_STATE* macro and the *__ratelimit* function - see the implementation of *printk_ratelimited* for an example.

Be sure to *#include * in order to use *printk_ratelimited()*

Both functions have also their *pr_** equivalents like* pr_info_ratelimited *for* printk_ratelimited(KERN_INFO... *and* pr_crit_once *for* printk_once(KERN_CRIT...*

Note: both did not work as expected in my tests here, will probably investigate further

Printk from userspace

Sometimes, especially when doing automated testing, it is quite useful to insert some messages in the kernel log buffer in order to annotate what's going on.

It is as simple as

```
# echo "Hello Kernel-World" > /dev/kmsg
```

Of course this messages gets the default log level assigned, if you want e.g. to issue a KERN_CRIT message you have to use the string representation of the log level - in this case "2"

```
# echo "2Writing critical printk messages from userspace" >/dev/kmsg
```

The message will appear like any other **kernel** message - there is **no way** to distinguish them!

Note: Don't confuse this with printf - we are printing a kernel message from userspace here.

If /dev/kmsg does not exist, it can be created with: 'mknod -m 600 /dev/kmsg c 1 11'

Internals / Changing the size of the printk buffer

Printk is implemented by using a ring buffer in the kernel with a size of *__LOG_BUF_LEN* bytes where *__LOG_BUF_LEN* equals *(1* \<\< CONFIG_LOG_BUF_SHIFT)*(see* kernel/printk.c for details).

You can specify the size of the buffer in your kernel config by setting *CONFIG_LOG_BUF_SHIFT* to an appropriate value (e.g. 17 for 128Kb) (*make menuconfig -> General Setup -> Kernel log buffer size*).

Using a ring buffer implies that older messages get overwritten once the buffer fills up, but this is only a minor drawback compared to the robustness of this solution (i.e. minimum memory footprint, callable from every context, not many resources wasted if nobody reads the buffer, no filling up of disk space/ram when some kernel process goes wild and spams the buffer, ...). Using a reasonably large buffer size should give you enough time to read your important messages before they are overwritten.

Note: dmesg reads by default a buffer of max 16392 bytes, so if you use a larger logbuffer you have to invoke dmesg with the *-s* parameter e.g.:

```
### CONFIG_LOG_BUF_SHIFT 17 = 128k
$ dmesg -s 128000
```

The kernel log buffer is accessible for reading from userspace by */proc/kmsg*. */proc/kmsg* behaves more or less like a FIFO and blocks until new messages appear.

Please note - reading from */proc/kmsg* consumes the messages in the ring buffer so they may not be available for other programs. It is usually a good idea to let *klogd* or *syslog* do this job and read the content of the buffer via *dmesg*.

Pros and Cons

The main advantage of printk over other debugging solutions is that it requires no sophisticated setup and can be called anywhere from any time. Printk can be called while holding a lock, from interrupt and process context, is SMP safe and does not need any special preparation. It is just there and just works. The only precondition is that you have some kind of working console to display the messages.

For the early stages in the boot process, where no console is available yet, there is a special function named *early_printk*, this function writes directly to the VGA buffer or a serial line but otherwise works just like printk -- you have to enable this function by setting *CONFIG_EARLY_PRINTK* in your kernel config (*make menuconfig -> Kernel Hacking -> Early printk*).

The major drawback is that printk is quite static, so you have to figure out what you want to trace beforehand and if you want to trace something different you have to recompile your code - which can become quite cumbersome. (And of course printk is not interactive at all, so you can't modify any variables or the like.)

The other drawback is that printing usually consumes quite some processing power and io time, so if you're trying to debug a timing critical section or a timing bug, you're probably out of luck.

Debugging early boot problems

Accessing the printk buffer after a silent hang on boot

Sometimes, if the kernel hangs early in the boot process, you get no messages on the console before the hang. However, there may still be messages in the printk buffer, which can give you an idea of where the problem is.

The kernel starts putting messages into the printk buffer as soon as it starts. They stay buffered there until the console code has a chance to initialize the console device (often the serial port for embedded devices). Even though these messages are not printed before the hang, it is still possible in some circumstances to dump the printk buffer and see the messages.

The tricky parts of doing this are:

1. using a warm reset (if possible) so the memory contents are not lost when transitioning from the stuck kernel to the bootloader. You *can* do a cold boot, but if the memory is left unpowered for very long, you will start to see memory corruption.
2. figuring out the address to use in the bootloader, based on the address of __log_buf in System.map. You will probably need to subtract the value of CONFIG_PAGE_OFFSET from the __log_buf address. However, there may be other offsets present as well (such as TEXT_OFFSET). Sometimes you can find the buffer by dumping the memory in a suspected area and locating the kernel messages visually in the dump. Note that the mapping offset between the kernel map of memory and the bootloader map of memory should not change. So once you figure it out you are set.

Quinn Jensen writes:

Something I've found handy when the console is silent is to dump the printk buffer from the boot loader. To do it you have to know how your boot loader maps memory compared to the kernel.

Redboot example on a Freescale ADS board

Quinn says: Here's what I do with Redboot on i.MX31:

```
fgrep printk_buf System.map
```

this shows the linked address of the printk_buf, e.g.:

```
c02338f0 b printk_buf.16194
```

The address "c02338f0" is in kernel virtual, which, in the case of i.MX31 ADS, redboot will have mapped to 0x802338f0. So, after resetting the target board, but without letting it try to boot again, at the redboot prompt:

```
dump -b 0x802338f0 -l 10000
```

And you see the printk buffer that never got flushed to the UART. Knowing what's there can be **very** useful in debugging your console.

U-boot example on an OMAP OSK board

Tim Bird tried these steps and they worked:

```
grep __log_buf System.map
```

or

```
grep __log_buf /proc/kallsyms
```

These show:

```
c0352d88 B __log_buf
```

In the case of the OSK, this address maps to 0x10352d88. So I reset the target board and use the following:

```
OMAP5912 OSK # md 10352d88
10352d88: 4c3e353c 78756e69 72657620 6e6f6973    <5>Linux version
10352d98: 362e3220 2e32322e 612d3631 6e5f706c     2.6.22.16-alp_n
10352da8: 7428206c 64726962 6d697440 6b736564    l (tbird@timdesk
10352db8: 2e6d612e 796e6f73 6d6f632e 67282029    .am.sony.com) (g
10352dc8: 76206363 69737265 33206e6f 342e342e    cc version 3.4.4
10352dd8: 34232029 45525020 54504d45 65755420    ) #4 PREEMPT Tue
...
```

Grub

Grub also supports a dump command which you can invoke from the grub prompt.

```
dump  [ -s offset ] [-n length] { FILE | (mem) }
```

Using CONFIG_DEBUG_LL and printascii() (ARM only)

If the kernel fails before the serial console is enabled, you can use CONFIG_DEBUG_LL to add extra debug output routines to the kernel. These are printascii, printch and printhex. These routines print directly to the serial port, bypassing the console code, and are available earlier in machine initialization.

To see printks earlier in the boot sequence (before the console is initialized), set CONFIG_DEBUG_LL=y and CONFIG_EARLY_PRINTK=y.

Alternatively, add your own calls to printascii, printch, and printhex where you believe the problems are located.

Here is an e-mail exchange seen on the linux-embedded mailing list (with answer by George Davis):[3]

```
> When we try to boot a 2.6.21 kernel after uncompressing the kernel the boot process dies somehow.
> We've figured out so far that the kernel dies somewhere between the gunzip and start_kernel.

Try enabling DEBUG_LL to see if it's an machine ID error.  If you see:

Error: unrecognized/unsupported processor variant.

or:

Error: unrecognized/unsupported machine ID...

Then you either don't have proper processor support enabled for your target or your bootloader is passing in the wron
g machine number.

If you still don't see anything, try hacking printk.c to call printascii() (enabled for the DEBUG_LL case) to print d
irectly to the serial port w/o a driver, etc.,.  You can find more details on these low-level debugging hacks via a l
ittle googling...
```

NetConsole

Sometimes you are in the unlucky situation that the machine crashes or hangs and you have no way to access the console afterwards, e.g. the graphic driver hangs and and the kernel dies soon after. In this case you could either hook up a serial line to your crashing target machine (if a serial port is available) or use the kernels netconsole feature to enable printk logging via UDP.

In order to use it you have to enable the *CONFIG_NETCONSOLE* kernel config option (*make menuconfig -> Device Drivers -> Network device support -> Network core driver support -> Network console logging support*) and configure it appropriately by using the *netconsole* configuration parameter

```
netconsole=[src-port]@[src-ip]/[<dev>],[tgt-port]@<tgt-ip>/[tgt-macaddr]
    where
        src-port      source for UDP packets (defaults to 6665)
        src-ip        source IP to use (interface address)
        dev           network interface (eth0)
        tgt-port      port for logging agent (6666)
        tgt-ip        IP address for logging agent
        tgt-macaddr   ethernet MAC address for logging agent (broadcast)
```

e.g. specify at kernel commandline (in your bootloader)

```
linux netconsole=4444@10.0.0.1/eth1,9353@10.0.0.2/12:34:56:78:9a:bc
```

to send messages from 10.0.0.1 port 4444 via eth1 to 10.0.0.2 port 9353 with the MAC 12:34:56:78:9a:bc

or while loading the module e.g

```
# insmod netconsole netconsole=@/,@10.0.0.2/
```

to send messages via broadcast to 10.0.0.2 port 6666

On the other machine you can now simply fire up

```
# netcat -u -l -p <port>
```

e.g.

```
$ netcat -u -l -p 6666
```

and see the printk messages from your target dribbling in.

If you don't see any messages you might have to set the console_loglevel to a higher value (see above) or test the connection via telnet e.g. from the target type

```
$ telnet 10.0.0.2 6666
```

Netconsole resources

See Documentation/networking/netconsole.txt for more details.

See Sarah Sharp's blog entry about using netconsole

Misc

Dmesg / Clearing the buffer

```
dmesg -c
```

clears the dmesg buffer. Sometimes it is nice to start with a blank buffer, so you will only see new messages when you invoke *dmesg*

Printk Timestamps

```
CONFIG_PRINTK_TIME
```

Setting this kernel config option prepends every printk statement with a timestamp representing the time since boot. This is particularly useful to get a general idea about the timings of your code.

You can also specify an argument on the kernel command line to enable this, or you can enable it any time at runtime by doing the following:

```
 $echo 1 >/sys/module/printk/parameters/time
```

Also, there are tools available to use the information to show relative times between printks (scripts/show_delta) and create graphs of durations in the kernel (scripts/bootgraph.pl)

See Printk Times for more details

Printing buffers as hex

If you want to print a buffer as hex within the kernel, don't reinvent the wheel use *printk_hex_dump_bytes()* instead.

```
print_hex_dump_bytes(const char *prefix_str, int prefix_type, const void *buf, size_t len)
```

this function prints a buffer as hex values to the kernel log buffer (with level KERN_DEBUG) Example:

```
Kernel Code:
char mybuf[] = "abcdef";
print_hex_dump_bytes("", DUMP_PREFIX_NONE, mybuf, ARRAY_SIZE(mybuf));

dmesg output:
61 62 63 64 65 66 00                             abcdef.
```

If you need something more sophisticated and flexible maybe have a look at *print_hex_dump()* and *hex_dump_to_buffer()*

Dynamic Debugging

It is also possible to enable/disable debug information at runtime using the dynamic debug functionality of the kernel. For this the *CONFIG_DYNAMIC_DEBUG* kernel config option must be set. See Documentation/dynamic-debug-howto.txt for more information.

Disabling printk messages at compile time

There is a configuration option which allows you to turn off all the printk messages in the whole kernel (*CONFIG_PRINTK*). This reduces the size the kernel, usually by at least 100k, since all message strings are not compiled into the kernel binary image.

However, it also means you get absolutely no output from the kernel while it is running. Disabling kernel printk messages is usually the last thing you do when you are tuning your kernel for size.

References and external links

- Linux Kernel Development, Robert Love, 3rd Edition, Chapter 18 Debugging
- Linux Device Drivers, Corbet, Rubini and Kroah-Hartmann, 3rd Edition, Chapter 4 Section 2
- Essential Linux Device Drivers

- Documentation/printk-formats.txt
- Documentation/dynamic-debug-howto.txt
- Documentation/networking/netconsole.txt
- kernel/printk.c Implementation of printk and others
- include/linux/printk.h printk header file
- include/linux/kern_levels.h logging levels header file
- Blog Entry about the different %p format specifiers
- LWN.net: The perils of pr_info()
- Kernel logging: APIs and implementation, Tim Jones (for IBM) (nice article)

Some page related to printk:

- Printk Times - has information about how to turn on timestamps for each printk message
  - printk time stamps sample
- printk size information
- Do Printk - has information about a method of disabling printk messages on a per-module basis

Categories:

- Development Tools
- Tips and Tricks
- Printk

> From: eLinux.org

Kernel Debugging Tips

Here are some miscellaneous tips for debugging a kernel:

Contents

Using printks

To add your own debug message to the kernel, you can place a "printk()" in the kernel code.

See Debugging by printing -> Usage for more details.

Log levels

Each kernel message can be pre-pended with a tag indicating the importance of the message.

See Debugging by printing -> Log_Levels for more details.

Adding timing information

Sometimes, it is useful to add timing information to the printk values, so you can see when a particular event occurred. The kernel includes an feature for doing this called printk times.

See the help for CONFIG_PRINTK_TIMES in the file lib/Kconfig.debug for more information on this feature. This option is found on the "Kernel Hacking" menu when configuring the kernel.

The timestamps which are inserted into the printk output consist of seconds and microseconds, as absolute values from the start of machine operation (or from the start of kernel timekeeping).

There is also tool in the kernel source which will convert the timestamp values to relative values (so you can see the interval between events). This tools is called show_delta and is located in the kernel 'scripts' directory.

See Printk Times for more information.

Viewing log messages

The `klogd` program will extract messages from the kernel log buffer, and send them to the system log (which winds up in /var/log/messages on most systems). This command runs in the background on most desktop or server systems, and continually transfers messages from the kernel log buffer to the system log.

You can view the contents of the log buffer directly, using the `dmesg` command. Note that by default `dmesg` displays the messages from the buffer, but does not remove them. So this command can be run multiple times to view the kernel printk messages. See the dmesg man page for more things you can do with this tool.

Controlling console output

In order to have the kernel boot be less "noisy", or in order to boot more quickly, it is sometimes useful to control the amount of messages displayed to the console during boot. You can do this by setting the kernel log level at boot time via a kernel command line option. See the "loglevel=" argument in `Documentation/kernel-parameters.txt`.

You can turn off all messages using the kernel command line option "quiet". See Disable Console for information on how much time this can save at boot up.

Note that even if the log level is changed, or "quiet" is used, although the printk messages are not print to console, they are still entered into the log buffer, and they can still be extracted and displayed later using the `dmesg` command.

Changing the size of the printk buffer

See Debugging by printing -> Internals / Changing the size of the printk buffer

Using kernel symbols

You can look up the source code for a function address using your toolchain's addr2line program. See Find a kernel function line or Addr2line for kernel debugging.

Using a kernel debugger

You can use the in-kernel debugger: KDB

You can use the in-kernel remote debugger: Kgdb

Also, you can use QEMU and gdb (and a high-level IDE like eclipse).

See Debugging the Linux kernel using Eclipse/CDT and Qemu for a great article on using Eclipse (with the CDT plugin) to debug the Linux kernel.

Debugging early boot problems

See Debugging_by_printing#Debugging_early_boot_problems

Triggering a kernel event

Overloading the sync system call

Sometimes, it is nice to trigger an event to happen in the kernel from user space. Instead of creating infrastructure to handle a /proc event, an ioctl() or making a new syscall, it can be quick and easy to just overload an existing function. One function not used very often is sync. (I have found that the sync system call is not normally called by user space programs (or during standard linux booting).

It is quite easy to put a hook to your own kernel program in the sys_sync() routine (located in fs/sync.c) and cause it to execute by issuing 'sync' from the shell command line. This is handy as a temporary mechanism to test things that you have put in the kernel.

Interpreting an Oops message

When the kernel encounters an internal fault, it will print an Oops message. Here are some tips on using the Oops message to find the source of the problem.

- See Documentation/oops-tracing.txt
- See HOWTO find oops location by Denis Vlasenko

Compilation tricks for the kernel

Sometimes, you want to modify how the compiler builds an individual kernel file. The following are tips for doing tasks related to this.

Build an individual file

You can build an individual output object file, with:

```
make fs/buffer.o
```

This will build JUST fs/buffer.o (if it needs rebuilding) and not the entire kernel. To force it to need re-building, use 'touch' on the associated source file:

```
touch fs/buffer.c
```

Create the preprocessed file for an individual source file

Using the same technique, you can create the preprocessed file for a C source file. This is useful if you're having trouble tracking down macro expansion or where defines/prototypes are coming from exactly.

```
make fs/buffer.i
```

Create the assembly file for an individual source file

Using the same technique, you can create the assembly file for a C source file. This is useful to get an idea what actual machine instructions are generated from the C source code.

```
make fs/buffer.s
```

Another way to get the raw assembly, is to dump the object file using 'objdump'

```
objdump -d fs/buffer.o > fs/buffer.disassem
```

This will produce a disassembly of the object file, which should show how the assembly was translated into machine instructions.

If the object has been compiled with debug symbols (using '-g'), then you might get more information using the '-S' option with objdump:

```
objdump -S -d fs/buffer.o >fs/buffer.disassem
```

You can also request that the toolchain show mixed source and assembly, by passing extra flags:

```
make EXTRA_CFLAGS="-g -Wa,-a,-ad -fverbose-asm" fs/buffer.o >fs/buffer.mixed
```

Alter the flags for a compilation

Sometimes, you need to alter the compilation flags for an individual file. There are two ways to do this. One is to add the extra flags on the make command line:

```
make EXTRA_CFLAGS="-g -finstrument-functions" fs/buffer.o
```

This will work if the flags can be appended to the regular set of C flags used for compiling the object.

However, if you need to do something more complicated, like removing or modifying flags, then you can build your own command line by hand. To do this, it is easiest to have 'make' produce the default compilation command (which will be several lines long), then copy, paste and edit it, to run on the command line directly. To see the exact compile commands used to compile a particular object, use the V=1 option with the kernel build system:

```
make V=1 fs/buffer.o
```

For me, this produced something like this:

mipsel-linux-gcc -Wp,-MD,fs/.buffer.o.d -nostdinc -isystem /home/usr/local/mipsel-linux-glibc/bin/../lib/gcc/mipsisa32el-linux/3.4.3/include -D__KERNEL__ -Iinclude -Iinclude2 -I/home/tbird/work/linux/include -I/home/tbird/work/linux/fs -Ifs -Wall -Wstrict-prototypes -Wno-trigraphs -fno-strict-aliasing -fno-common -ffreestanding -O2 -fomit-frame-pointer -g -I/home/tbird/work/linux/ -I /home/tbird/work/linux/include/asm/gcc -G 0 -mno-abicalls -fno-pic -pipe -finline-limit=100000 -mabi=32 -march=mips32r2 -Wa,-32 -Wa,-march=mips32r2 -Wa,-mips32r2 -Wa,--trap -I/home/tbird/work/linux/include/asm-mips/ati -Iinclude/asm-mips/ati -I/home/tbird/work/linux/include/asm-mips/mach-generic -Iinclude/asm-mips/mach-generic -Wdeclaration-after-statement -DKBUILD_BASENAME=buffer -DKBUILD_MODNAME=buffer -c -o fs/buffer.o /home/tbird/work/linux/fs/buffer.c

Category:

- Development Tools

> From: eLinux.org

Kgdb

It is fascinating to think that you have control over running Linux Kernel. You can stop, can single-step, can resume and even can put break-points on running Kernel. In fact, you can debug the kernel as easily as you debug any application.

Contents

How to setup kgdb

The steps mentioned here are with reference to 2.6.26 Kernel. The main reason is KGDB code is merged into Linux tree from 2.6.26-RC5 kernel. (As a side note, for kernel \< 2.6.26-RC5, you have to get kgdb patch from [http://kgdb.linsyssoft.com/kernel.htm] and apply them to kernel)

Hardware Requirements

Two x86 machines are required for using KGDB. One of the machines runs a kernel to be debugged called "TEST MACHINE". The other machine runs gdb "DEVELOPMENT MACHINE". A serial line is required between the development and the test machine. And so obviously, machines need one serial port each. Basically, you will be sending "Debugging Commands" from "DEVELOPMENT MACHINE" to "TEST MACHINE".

Preparing Kernel to be Debugged

1. Download the source of kernel (for e.g., 2.6.26.2)from kernel.org

2. Recompile the Kernel on "DEVELOPMENT MACHINE". Go to Kernel Hacking and Enable the following options:

   -- *Magic SysRq key []* Compile the kernel with debug info [*] KGDB: kernel debugging with remote gdb --->

And in KGDB:

```
--- KGDB: kernel debugging with remote gdb
<*>   KGDB: use kgdb over the serial console
[ ]   KGDB: internal test suite
```

1. Build kernel and modules

   ```
   make -j 12 && make modules && make modules_install
   ```

2. Transfer the vmlinux and system.map and initrd.img files on "TEST MACHINE".

3. Now, edit the GRUB entry for that kernel on "TEST MACHINE". Add kernel options/kernel parameters like kgdbwait and kgdboc=ttyS0,115200

```
For e.g., kernel /vmlinux ro root=LABEL=/ rhgb quiet crashkernel=128M@16M kgdbwait kgdboc=ttyS0,115200
```

kgdbwait -- > This will make Kernel to wait on boot time and will expect someone to connect to it and give further commands kgdboc --> This is a KGDB I/O driver and we are supplying two arguments. ttyS0 will tell that communication will happen on Serial Port 0 and 115200 is the baudrate.

1. Now boot the Kernel with those kernel parameters.
2. On dev machine, start GDB session.

```
[dev@einfochips]gdb vmlinux
```

The argument vmlinux file is the file that is created with Debug symbols. It will be of much larger size and more than likely to be in the directory where you gave "make" command..

1. Assuming that on "DEVELOPMENT MACHINE" you have set serial interface baudrate as 115200. Connect to the "TEST MACHINE" with target command.

   ```
   (gdb)target remote /dev/ttyS0
   ```

2. This will stop your Kernel booting on "TEST MACHINE" and will give control to your "DEVELOPMENT MACHINE". Now, you can do Single-stepping or put breakpoints and etc.
3. Once your Kernel is running on "TEST MACHINE" and you want control over your running kernel from "DEVELOPMENT MACHINE", You have to send MANUALLY on TEST machine SysRq command. So, on "TEST MACHINE" press SysRq + g

   (i.e., press "ALT" key then Press "PrintScreen" Key and then Press "g" key)

There are several possible problems that you may face

1. Your Kernel is booted and SysRq+g is not working.

```
[r00t@einfochips] echo 1 > /proc/sys/kernel/sysrq
```

This will enable sending SysRq commands.

1. You may find some time that while stopping execution through SysRq key on "TEST MACHINE", it stops but then it is not able to communicate over serial cable with "DEVELOPMENT MACHINE". The reason can be, your KGDB I/O driver is not passed arguments properly and you may need to reconfigure the driver by following way,

   ```
   [root@einfochips]echo "ttyS0,115200" > /sys/modules/<module name>/parameters
   ```

2. The target board has a single serial port that needs to be shared between the console and kgdb.

```
The kgdb Docbook refers to using a proxy to split the serial port data stream
between gdb and the console terminal emulator.  proxy programs include:

  kgdb demux (kdmx) Media:Kdmx_v140904a.tar.gz
                     (moved to agent-proxy git)
  kgdb demux (kdmx)
                     kdmx at elinux.org
                     https://git.kernel.org/cgit/utils/kernel/kgdb/agent-proxy.git/
                     git clone git://git.kernel.org/pub/scm/utils/kernel/kgdb/agent-proxy.git/
  agent-proxy
                     https://git.kernel.org/cgit/utils/kernel/kgdb/agent-proxy.git/
                     git clone git://git.kernel.org/pub/scm/utils/kernel/kgdb/agent-proxy.git/

(frowand: I have been using kdmx successfully.  I have not tried using agent-proxy.)
```

Using an IDE

You can debug Linux Kernel over KGDB with Visual Studio using the VisualKernel plugin. This tutorial shows how to use it with KGDB.

Category:

- Development Tools

KDB

Contents

Introduction and basic resources

Here is some information about KDB - the in-kernel debugger for the Linux kernel.

The KDB and KGDB official wiki: https://kgdb.wiki.kernel.org/ (this only has 2 pages?)

Jason Wessel is the current KDB maintainer. Here is a presentation from him at LinuxCon 2010 (August 2010): http://kernel.org/pub/linux/kernel/people/jwessel/dbg_webinar/State_Of_kernel_debugging_LinuxCon2010.pdf

Here are some videos showing use of KDB and KGDB:

- video 1 of 6: http://www.youtube.com/watch?v=V6Qc8ppJ_jc
  - example of a call to panic from a test module (without a debugger)
- video 2 of 6: http://www.youtube.com/watch?v=LqAhY8K3XzI
  - example of catching the panic with KDB, and looking up the source line with gdb
- video 3 of 6: http://www.youtube.com/watch?v=bBEh_UduX04
  - example of a bad access request, and looking up the source line with gdb
- video 4 of 6: http://www.youtube.com/watch?v=MfJU2E0aJwg
  - example of using a hardware breakpoint with kdb
- video 5 of 6: http://www.youtube.com/watch?v=sWiHV5mt8_k
  - use an address watch (hardware watchpoint) using kgdb (data access hardware breakpoint on tp_address_ref)
- video 6 of 6: http://www.youtube.com/watch?v=nnopzcwvLTs

- use of kgdb over serial - Start up the agent-proxy and connect and hit a breakpoint a sys_sync

Documentation, up-to-date as of 2010, for KDB and KGDB is at: http://kernel.org/pub/linux/kernel/people/jwessel/kdb/

Older Information

See http://www.ibm.com/developerworks/linux/library/l-kdbug/ for a tutorial for the 2.4.20 kernel (from June 2003)

Here's an article from 2002 on KDB vs. KGDB: http://kerneltrap.org/node/112 It has a good discussion excerpt between Andrew Morton and Keith Owens about the relative merits of KDB versus KGDB.

General Information

Kernel Versions supported

kgdb was added to the mainline Linux kernel in version 2.6.26.

kdb support was added to the mainline Linux kernel in version 2.6.35.

Before those versions, kgdb and kdb were available as patches which could be applied to the Linux kernel source.

Kernel configuration

The following descriptions are for a 2.6.35 kernel, using KDB over a serial line between host and target:

All these options on are the "Kernel Hacking" menu.

In order to support KDB, "KGDB" support must be turned on first (even if you aren't using kgdb/gdb)

Config variables

- CONFIG_DEBUG_KERNEL=Y - includes debug information in the kernel compilation - required for basic kernel debugging support
- CONFIG_KGDB=Y - turn on basic kernel debug agent support
- CONFIG_KGDB_SERIAL_CONSOLE=Y - to share a serial console with kgdb.
  - Sysrq-g must be used to break in initially.
  - Selecting this will automatically set:
    - CONFIG_CONSOLE_POLL=Y
    - CONFIG_MAGIC_SYSRQ=Y - turn on MAGIC-SYSRQ key support
- CONFIG_KGDB_KDB=Y - actually turn on the KDB debugger feature
- CONFIG_DEBUG_RODATA=N - you must disable this on x86 in order to set breakpoints on code or data

Optional other configuration settings:

- CONFIG_FRAME_POINTER=Y - this allows for better backtrace support in KDB
- CONFIG_KALLSYMS=Y - this adds symbolic information to the kernel, useful to see symbols instead of addresses
- CONFIG_KDB_KEYBOARD - use KDB with an attached keyboard (not for use with serial console)
- CONFIG_KGDB_TESTS - used to turn on kgdb internal self-tests - see the config help for this for more information

Steps

To turn on KDB over serial console, do the following:

- 'make menuconfig'
  - go to "Kernel Hacking" sub-menu
  - turn on "KGDB: kernel debugger", and choose "\" to go to sub-menu
  - verify that "KGDB: use kgdb over the serial console" is set

  - set "KGDB_KDB: include kdb frontend for kgdb"
- save and exit

Results:

```
[]$ diffconfig
KGDB n -> y
+CONSOLE_POLL y
+KDB_KEYBOARD n
+KGDB_KDB y
+KGDB_SERIAL_CONSOLE y
+KGDB_TESTS n
```

Enabling kdb

At runtime

Once the kernel is compiled with kdb support and is running on your target board, you need to enable it. This can be done on a running system, binding the kdb/kgdb feature to a serial port, by writing a value into the sys filesystem.

If your machine starts a serial console on ttyS0, you can bind kdb/kgdb to this serial console by writing the string "ttyS0" to /sys/module/kgdboc/parameters/kgdboc. The kernel will respond with a message indicating that that driver is registered.

```
$ echo ttyS0 >/sys/module/kgdboc/parameters/kgdboc
kgdb: Registered I/O driver kgdboc.
```

At boot time (via kernel command line)

You can enable kdb support in your kernel at boot time by using the 'kgdboc' option on the kernel command line. Normally, you specify the tty device name, followed by the serial port speed.

```
kgdboc=ttyS0,115200
```

Invoking kdb

Once the kernel is running, and the kgdb/kdb is bound to the serial console, you can invoke the debugger in numerous ways.

First, you can enter the debugger using a Magic SysRq command.

kdb will also be entered automatically if the kernel panics.

Finally, you can set a breakpoint (either hardware or software), such that the debugger is invoked when the breakpoint condition is met. For a code breakpoint, this means when the instruction is executed at the breakpoint location, and for a data breakpoint, when the particular access is made at the breakpoint address.

Invoking with Magic SysRq 'g'

To invoke the debugger using the Magic SysRq command, you use the 'g' command, which can be issued any of the ways supported by the Magic SysRq feature. This can be done by 1) typing the key sequence on a connected keyboard, 2) echoing a value to /proc/sysrq-trigger, or 3) sending a break key sequence through the serial console

Examples:

- sysrq trigger from local shell, via procfs: 'echo g >/proc/sysrq-trigger'
- sysrq trigger via serial console break sequence:

- in minicom, type 'ctrl-a f g' (quickly)
- in telnet, through a server that supports sending a break: type

Invocation by panic

When a kernel panic occurs, then something has gone seriously wrong and the kernel automatically enters kdb. From here you can look at memory, do a traceback, examine registers, and do other operations to find out more about the state of the system and debug the problem.

Invocation by breakpoint

To enter kdb using a breakpoint, first invoke kdb using the Magic SysRq key (see above), then set a breakpoint. Then type 'go' to continue execution. When the breakpoint is hit, the debugger shell will appear.

In the example that follows, items in italics are commands typed by a user. Items following a '$' are commands entered at a shell command (normal Linux user-space runtime), and items following 'kgdb>' are commands entered at the kdb interactive shell.

```
$ echo g >/proc/sysrq-trigger
SysRq : DEBUG

Entering kdb (current=0xdfdff040, pid 71) due to Keyboard Entry
kdb> bp sys_sync+4
Instruction(i) BP #0 at 0xc00c9f00 (sys_sync+0x4)
   is enabled  addr at 00000000c00c9f00, hardtype=0 installed=0

kdb> go
$ sync

Entering kdb (current=0xdfdaa360, pid 72) due to Breakpoint @ 0xc00c9f00
kdb> bt
Stack traceback for pid 72
0xdfdaa360       72       71  1    0   R  0xdfdaa560 *sync
[<c0028cb4>] (unwind_backtrace+0x0/0xe4) from [<c0026d50>] (show_stack+0x10/0x14)
[<c0026d50>] (show_stack+0x10/0x14) from [<c0079e78>] (kdb_show_stack+0x58/0x80)
[<c0079e78>] (kdb_show_stack+0x58/0x80) from [<c0079f1c>] (kdb_bt1.clone.0+0x7c/0xcc)
[<c0079f1c>] (kdb_bt1.clone.0+0x7c/0xcc) from [<c007a240>] (kdb_bt+0x2d4/0x338)
[<c007a240>] (kdb_bt+0x2d4/0x338) from [<c0078328>] (kdb_parse+0x4d4/0x5f8)
[<c0078328>] (kdb_parse+0x4d4/0x5f8) from [<c0078a8c>] (kdb_main_loop+0x448/0x6b0)
[<c0078a8c>] (kdb_main_loop+0x448/0x6b0) from [<c007acb4>] (kdb_stub+0x210/0x398)
[<c007acb4>] (kdb_stub+0x210/0x398) from [<c0073280>] (kgdb_handle_exception+0x384/0x574)
[<c0073280>] (kgdb_handle_exception+0x384/0x574) from [<c0028518>] (kgdb_brk_fn+0x18/0x20)
[<c0028518>] (kgdb_brk_fn+0x18/0x20) from [<c0022198>] (do_undefinstr+0x10c/0x1a8)
[<c0022198>] (do_undefinstr+0x10c/0x1a8) from [<c0022b84>] (__und_svc+0x44/0x60)
Exception stack(0xdfe09f58 to 0xdfe09fa0)
9f40:                                                       00000000 bec93e74
9f60: 000437ac 00034738 00000001 bec93e74 00000049 00000024 c00230e8 dfe08000
9f80: 00000000 bec93e54 00000000 dfe09fa0 c0022f40 c00c9f00 80000013 ffffffff
[<c0022b84>] (__und_svc+0x44/0x60) from [<c00c9f00>] (sys_sync+0x4/0x28)
[<c00c9f00>] (sys_sync+0x4/0x28) from [<c0022f40>] (ret_fast_syscall+0x0/0x30)
kdb>
```

This example shows an invocation of kdb, followed by setting a breakpoint, then resuming execution with 'go'. Then, at the Linux user-space shell, the 'sync' command is run to cause the breakpoint to occur. When kdb is entered due to the breakpoint, then 'bt' is run to get a backtrace from the stack of the current process.

Using gdb to see the kernel source listing

You can use the addresses printed out in kdb, with a host-side gdb session, to see the source code or assembly instructions around a particular address.

The target address can come from a backtrace or register dump (e.g. instruction pointer).

To load the source for a kernel, start gdb (or the appropriate arch-specific gdb) with the vmlinux that matches the image running on target. The kernel should have been compiled with debug symbols (CONFIG_DEBUG_KERNEL=y). gdb will start, and load the symbol information for the kernel.

Use the following commands to see various bits of information:

- source file and line number for an instruction address
  - *info line* \0x*
- source lines around an instruction address
  - *list* \0x*
- assembly instructions at an address
  - *disas 0x*, or
  - *x/20i 0x*

KDB environment variables

Here are some environment variables supported by kdb (in 2.6.35):

- LINES - set the number of lines for paging output from KDB (default 24)
- BTAPROMPT - prompt after each process in bta command (default 0)
- LOGGING - if 1, echo all KDB output to the kernel log buffer (printk log) (default 0)
- PROMPT - string to use for kdb interactive shell prompt (default '[%d]kdb>' on SMP or 'kdb>' on UP)
- MOREPROMPT - string to use for kdb paged output prompt (default '[%d]more>' on SMP or 'more>' on UP)
- RADIX - base used to print out (default 16)
- MDCOUNT - number of lines of data for md command if not specified (default 8)
- BYTESPERWORD - number of bytes per word for md command (default 4)
- DTABCOUNT - number of symbols to display for tab completion (default 30)
- NOSECT - avoid displaying section information with md commands (default 1)
- PS - specify a list of task states (one letter per state) for filtering the 'ps' command (default 'DRSTCZEU')
  - letters can be used from the list: DRSTCZEUIMA, each letter corresponding to a task state (see the PS man page for task states)
    - D = uninterruptible sleep
    - R = running
    - S = interruptible sleep
    - T = stopped
    - C = traced
    - Z = exited and zombied
    - E = exited and dead
    - U = unrunnable
    - I = idle
    - M = daemon
    - A = all

Unused args (in 2.6.35):

- BTARGS - number of arguments to show in bt command (default 9)
  - is used for argcount for internal routine kdb_bt1, but argcount is never used in that routine
- BTSYMARG - ??? [set in kdb_cmds 'dumpall' and 'dumpcpu', but not used in code)
  - appears to be an integer

Hints and tips

Command line completion

kdb supports command line completion using the tab key. While typing at the interactive shell, you can press the tab key to get a list of symbols to use as part of the command. If you type the first few letters of a symbol name, then press the tab key, the shell will complete the name for you. If more than one symbol matches what you have typed, then a list is shown of matching symbols. The number of symbols shown is controlled by the DTABCOUNT variable.

Defining a set of commands

You can define a set of commands to be run as a group (in sequence), using the 'defcmd' command. This group essentially becomes a new command which you can run, using the indicated name.

The second two strings after the new command name are the Usage string and the Description string. These appear in the online help when the 'help' or '?' commands are run.

Here is an example of the definition of a new command:

```
defcmd dumpcommon "" "Common kdb debugging"
  set BTAPROMPT 0
  set LINES 10000
  -summary
  -cpu
  -ps
  -dmesg 600
  -bt
endefcmd
```

A leading dash indicates to ignore any errors processing a command.

Executing commands on kdb initialization

In the kernel source, you can place commands in the file 'kdb_cmds'. These commands will be executed by kdb on it's initialization. By default, a few command sets are declared: 'dumpcommon', 'dumpall', and 'dumpcpu'.

Piping command results through 'grep'

kdb includes a grep-like capability, which provides the ability to filter command results using a pattern. To use this, type the regular command, and at the end of the line add '| grep \', where pattern is the pattern to match.

To see online help for this feature, type 'grephelp' at the kdb shell prompt.

The pattern may include a very limited set of metacharacters:

```
  pattern or ^pattern or pattern$ or ^pattern$
```

And if there are spaces in the pattern, you may quote it:

```
  "pat tern" or "^pat tern" or "pat tern$" or "^pat tern$"
```

Command history

You can access the command history by typing up-arrow and down-arrow, and select a previous command to run.

If you hit \ without typing a command, sometimes (when it makes sense) the last command run will be run again. For 'md' commands, the next command will continue at the address where the last command left off.

Feature extension notes (by Tim)

This section just has some random notes on Tim Bird's investigation of KDB (for kernel version 2.6.35)

Regular users of KDB can ignore this...

- Can break on __do_user_fault, to break into kdb on user process SIGSEGV (on ARM, anyway)
- See CONFIG_DEBUG_USER (on ARM) for extra information shown and code paths used, on user-space faults

- I can't set a breakpoint from kdb_cmds (on kdb initialization)
    - I get a hang (likely a panic before the console starts) on any breakpoint set on kdb initialization
        - how to debug: 1) printk and logbuf visibility over reset? 2) use of qemu??

internals

internal functions to retrieve environment variables:

- kdbgetintenv()
- kdbgetenv()
- kdbgetulenv() - get unsigned long env variable

Category:

- Development Tools

> From: eLinux.org

Kdmx

Contents

overview

If the target board has a single serial port then a proxy on the host is needed to be share the target port between the console and kgdb.

kdmx (kgdb demux) is one of the available proxy programs.

source location

The source is located in the agent-proxy git

```
https://git.kernel.org/cgit/utils/kernel/kgdb/agent-proxy.git/
git clone git://git.kernel.org/pub/scm/utils/kernel/kgdb/agent-proxy.git/
```

example 1 (simple)

This is a simple example, where I am using a USB serial port to connect between my host and target.

I did all of the commands in "host window 1" before I did the commands in "host window 2".

I did all of the commands in "host window 2" before I did the commands in "host window 3".

```
+---------------------  HOST  ------------------------------------+                +--  TARGET  --+
|                                                                  |                |              |
|                                                                  |                |              |
v                                                                  v                v              v

terminal  <---->  /dev/pty/29  <---->  +------+
emulator                               |      |
                                       | kdmx |  <---->  serial port  <== cable ==> target console
gdb  <--------->  /dev/pty/53  <---->  |      |          /dev/ttyUSB0
                                       +------+
```

The absolute path of the kernel source directory is in ${LINUX_SOURCE}.

The path of the kernel build directory, relative to the kernel source directory, is in ${KBUILD_OUTPUT}

```
$ echo ${LINUX_SOURCE}
/xxx/linux--3.17

$ echo ${KBUILD_OUTPUT}
../build/dragon_linus_3.17

$ echo ${LINUX_SOURCE}/${KBUILD_OUTPUT}
```

```
/xxx/linux--3.17/../build/dragon_linus_3.17

-----  host window 1 - kdmx  -----

$ export PS1='[1]: '
[1]: ./kdmx -n -d -p/dev/ttyUSB0 -b115200
serial port: /dev/ttyUSB0
Initalizing the serial port to 115200 8n1
/dev/pts/29 is slave pty for for terminal emulator
/dev/pts/53 is slave pty for gdb

-----  host window 2 - terminal emulator  -----

# In this example, I am using the sysrq debug command instead of the
# kernel command line 'kgdbwait' option to enter kgdb.
#
# The minicom "-o" option tells it not to initialize, so it does not
# send the "Init string" (defaults to "At S7=45 S0=0...") when you run
# minicom.

$ export PS1='[2]: '
[2]: minicom -o -w -p /dev/pts/29
Welcome to minicom 2.5

OPTIONS: I18n
Compiled on May  2 2011, 10:05:24.
Port /dev/tty8

Press CTRL-A Z for help on special keys

# Now in minicom, connected to the target console.
# CONFIG_PRINTK_TIME=y, so timestamps will be added to
# printk() messages to the console.

$ export PS1='% '
% cat /proc/version
Linux version 3.17.0-dirty (frank@ussvlx) (gcc version 4.6.x-google 20120106 (prerelease) (GCC) ) #1 SMP PREEMPT Mon
Oct 6 10:19:37 PDT 2014
% echo g >/proc/sysrq-trigger
[   24.246292] SysRq : DEBUG
[   24.247184] Entering KGDB

-----  host window 3 - gdb  -----

$ export PS1='[3]: '
[3]: echo ${CROSS_COMPILE}
arm-eabi-
[3]: cd ${LINUX_SOURCE}
[3]: strings $KBUILD_OUTPUT/vmlinux | grep "^Linux version"
Linux version 3.17.0-dirty (frank@ussvlx) (gcc version 4.6.x-google 20120106 (prerelease) (GCC) ) #1 SMP PREEMPT Mon
Oct 6 10:19:37 PDT 2014
Linux version
[3]: ${CROSS_COMPILE}gdb $KBUILD_OUTPUT/vmlinux
GNU gdb (GDB) 7.3.1-gg2
Copyright (C) 2011 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=x86_64-linux-gnu --target=arm-linux-android".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /xxx/git_linus/build/dragon_linus_3.17/vmlinux...done.
(gdb) target remote  /dev/pts/53
Remote debugging using /dev/pts/53
kgdb_breakpoint ()
    at /xxx/git_linus/linux--3.17/kernel/debug/debug_core.c:1050
1050            arch_kgdb_breakpoint();
(gdb) b sys_sync
Breakpoint 1 at 0xc031d450: file /xxx/git_linus/linux--3.17/fs/sync.c, line 103.
(gdb) c
Continuing.
```

example 2 (automate paths of gdb pty and terminal emulator pty)

Keeping track of the paths of the pty slaves created by kdmx can be annoying, so the status file feature has been added in kdmx version 141210a to automate use of the paths by gdb and the terminal emulator.

The status file feature is enabled with the '-s' option, which will create files containing the paths of the gdb slave pty and the terminal emulator pty. Each time a slave pty is opened, the contents of the respective status file is updated. The '-s' option provides the path, including a root for the file name, for the location of the status files. The strings '_gdb' and '_trm' will be appended to the path to form the names of the status files.

The absolute path of the kernel source directory is in ${LINUX_SOURCE}.

The path of the kernel build directory, relative to the kernel source directory, is in ${KBUILD_OUTPUT}

```
$ echo ${LINUX_SOURCE}
/xxx/linux--3.18

$ echo ${KBUILD_OUTPUT}
../build/dragon_linus_3.18

$ echo ${LINUX_SOURCE}/${KBUILD_OUTPUT}
/xxx/linux--3.18/../build/dragon_linus_3.18
```

In this example, I place the status files in my kernel build directory.

```
-----  host window 1 - kdmx  -----

$ export PS1='[1]: '
[1]: kdmx -n -d -p/dev/ttyUSB0 -b115200 -s${LINUX_SOURCE}/${KBUILD_OUTPUT}/zzz__kdmx_pty
gdb status file: /xxx/linux--3.18/../build/dragon_linus_3.18/zzz__kdmx_pty_gdb
terminal emulator status file: /xxx/linux--3.18/../build/dragon_linus_3.18/zzz__kdmx_pty_trm
serial port: /dev/ttyUSB0
Initalizing the serial port to 115200 8n1
/dev/pts/44 is slave pty for terminal emulator
/dev/pts/45 is slave pty for gdb

Use <ctrl>C to terminate program
```

Here are the contents of the status files:

```
$ ls -l ${LINUX_SOURCE}/${KBUILD_OUTPUT}/zzz__kdmx_pty*
-rw-r----- 1 frank frank_group 12 Dec 10 22:07 /xxx/linux--3.18/../build/dragon_linus_3.18/zzz__kdmx_pty_gdb
-rw-r----- 1 frank frank_group 12 Dec 10 22:07 /xxx/linux--3.18/../build/dragon_linus_3.18/zzz__kdmx_pty_trm
$ for k in `ls ${LINUX_SOURCE}/${KBUILD_OUTPUT}/zzz__kdmx_pty*`; do echo "$k:" ; cat $k; echo ; done
/xxx/linux--3.18/../build/dragon_linus_3.18/zzz__kdmx_pty_gdb:
/dev/pts/45

/xxx/linux--3.18/../build/dragon_linus_3.18/zzz__kdmx_pty_trm:
/dev/pts/44
```

The terminal emulator slave pty path is retrieved from the proper status file and provided to minicom:

```
-----  host window 2 - terminal emulator  -----

$ export PS1='[2]: '
[2]: minicom -o -w -p `cat ${LINUX_SOURCE}/${KBUILD_OUTPUT}/zzz__kdmx_pty_trm`
Welcome to minicom 2.5

OPTIONS: I18n
Compiled on May  2 2011, 10:05:24.
Port /dev/tty8

Press CTRL-A Z for help on special keys

$ export PS1='% '
% cat /proc/version
Linux version 3.18.0-dirty (frank@ussvlx) (gcc version 4.6.x-google 20120106 (prerelease) (GCC) ) #1 SMP PREEMPT Mon
Dec 8 12:27:26 PST 2014
% echo g > /proc/sysrq-trigger
[210227.044514] SysRq : DEBUG
```

The task of getting the path of the gdb slave pty into gdb is a little bit more complicated. (Any suggestions of a more straightforward method are welcome.)

The following shell script is used to invoke the proper gdb and create a user defined gdb command to connect to the correct pty path. The ${CROSS_COMPILE} ensures that the gdb built for the target architecture is used.

```
#! /bin/bash

kgdb_set_pty_cmd=`mktemp --tmpdir kgdb_set_pty_cmd.XXXXXXXXXX` || exit 1


# user defined command
#
#   \x5c is '\'
#
echo -e \
   "define tr"                                                              \
   "\ndont-repeat"                                                          \
   "\necho target remote"                                                   \
   "`cat ${LINUX_SOURCE}/${KBUILD_OUTPUT}/zzz__kdmx_pty_gdb`"               \
   "\x5cn"                                                                  \
   "\ntarget remote"                                                        \
   `cat ${LINUX_SOURCE}/${KBUILD_OUTPUT}/zzz__kdmx_pty_gdb`                 \
   "\nend"                                                                  \
   "\n"                                                                     \
   "\ndocument tr"                                                          \
   "\ntarget remote to the kdmx gdb pty"                                    \
   "\n"                                                                     \
   "\nThe kdmx gdb pty to be used is: "                                     \
   `cat ${LINUX_SOURCE}/${KBUILD_OUTPUT}/zzz__kdmx_pty_gdb`                 \
   "\nThe kdmx gdb pty was determined at gdb startup.  If the kdmx gdb pty"    \
   "\nhas changed since then, restart gdb"                                  \
   "\n"                                                                     \
   "\nend"                                                                  \
   > ${kgdb_set_pty_cmd}



echo "${CROSS_COMPILE}gdb -x ${kgdb_set_pty_cmd} $KBUILD_OUTPUT/vmlinux"
      ${CROSS_COMPILE}gdb -x ${kgdb_set_pty_cmd} $KBUILD_OUTPUT/vmlinux


rm ${kgdb_set_pty_cmd}

-----  host window 3 - gdb  -----

$ export PS1='[3]: '
[3]: echo ${CROSS_COMPILE}
arm-eabi-
[3]: cd ${LINUX_SOURCE}
```

```
[3]: tcgdb
arm-eabi-gdb -x /tmp/kgdb_set_pty_cmd.4xKNf2ah0T ../build/dragon_linus_3.18/vmlinux
GNU gdb (GDB) 7.3.1-gg2
Copyright (C) 2011 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=x86_64-linux-gnu --target=arm-linux-android".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /local/nobackup/src/git_linus/build/dragon_linus_3.18/vmlinux...done.
(gdb) help tr
target remote to the kdmx gdb pty

The kdmx gdb pty to be used is:  /dev/pts/45
The kdmx gdb pty was determined at gdb startup.  If the kdmx gdb pty
has changed since then, restart gdb

(gdb) tr
target remote /dev/pts/45
kgdb_breakpoint () at /local/nobackup/src/git_linus/linux--3.18/kernel/debug/debug_core.c:1050
1050        arch_kgdb_breakpoint();
(gdb) b sys_sync
Breakpoint 1 at 0xc031f7b4: file /local/nobackup/src/git_linus/linux--3.18/fs/sync.c, line 103.
(gdb) c
Continuing.
```

> From: eLinux.org

Debugging The Linux Kernel Using Gdb

Contents

Debugging the linux kernel using gdb

The majority of day to day kernel debugging is done by adding print statements to code by using the famous printk function. This technique is well described in Kernel Debugging Tips. Using printk is a relatively simple, effective and cheap way to find problems. But there are also many other Linux-grown techniques that take debugging and profiling to a higher level. On this page, we will discuss using the GNU debugger (GDB) to do kernel debugging. The GDB page describes some basic gdb command and also gives good links to documentation. Overall, starting using gdb to do kernel debugging is relatively easy.

Most of the examples here will work in two (open source) situations. When using JTAG and when using QEMU system emulation. As the second option does not require any hardware, you can go on and try it right away!

The open source JTAG debugging world is not that big. One project that stands out in terms of debugging capabilities is OpenOCD and this is the tool used in this documentation. OpenOCD is pretty usable on the ARM11 and ARM9 targets we tested.

Requirements

GDB:

You need to get a GDB that is capable of understanding your target architecture. Often, this comes with your cross-compiler, but if you have to, you can compile it yourself, but you need to understand the difference between --target and --host configure options. GDB will be running on the host (e.g. x86) but will be able to understand the target (e.g. armv6). You may also want to have the gdbserver that can serve as a stub for your user land debugging.

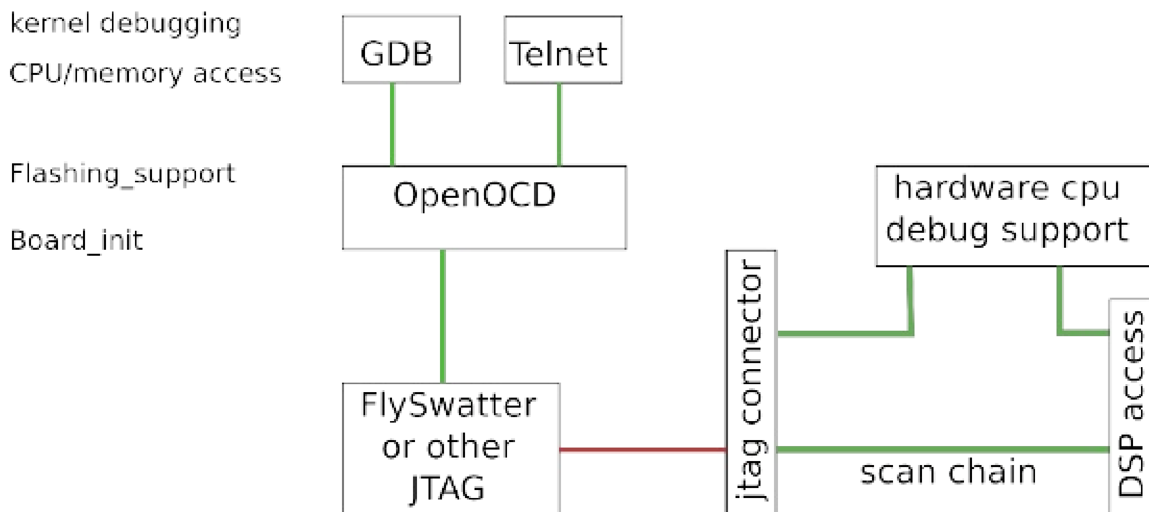
OpenOCD:

TODO...

A JTAG Dongle:

TODO...

The basics



To debug the kernel, you will need to configure it to have debug symbols. Once this is done, you can do your normal kernel development. When needed, you can "hook-up" your debugger. Start debugging a running kernel.

- start openocd

vmlinuz v.s zImage

When you want to debug the kernel you need a little understanding of how the kernel is composed. Most important is the difference between your vmlinux and the zImage. What you need to understand at this point is that the zImage is a container. This container gets loaded by a boot loader and that execution is handed over to the zImage. This zImage unpacks the kernel to the same memory location and starts executing the kernel. (explain that vmlinux does not have to be the real kernel as it is possible to debug a "stripped" kernel using a non stripped vmlinux). overall if we look at a compiled kernel we will see that vmlinux is located at the root of the kernel tree whiles the zImage is located under arch/arm/boot

```
vmlinux
arch/arm/boot
`-- zImage
```

vmlinux is what we will be using during debugging of the Linux kernel.

Debugging the kernel

The JTAG based debugging method described here is not intrusive. This means that besides debugging symbols you don't need to modify the kernel in any way. This is because we operate on the hardware, CPU core level. Overall this means that you can follow your normal development method. You can let your bootstrap and boot loader do their work and for example start debugging a running kernel. If your gdb-aware debugger is running it can be as simple as loading the vmlinuz and connecting to the remote target:

```
load vmlinuz
target remote :3333
```

Loading a kernel in memory

Once you are used to using gdb to debug kernels you will want to use gdb to directly load kernels onto your target. The most practical way of doing this is to set a hardware breakpoint at the start of the kernel and reset your board using the JTAG reset signal. Your boot loader will initialize your board and the execution will stop at the start of the kernel. After that you can load a kernel into memory and run it.

Execute the following:

```
(gdb) file vmlinux
(gdb) target remote :3333
(gdb) break __init_begin
(gdb) cont
(gdb) mon reset #perhaps this needs to be done from the openocd telnet session..
Breakpoint 1, 0xc0008000 in stext ()
(gdb) load vmlinux
Loading section .text.head, size 0x240 lma 0xc0008000
Loading section .init, size 0xe4dc0 lma 0xc0008240
Loading section .text, size 0x219558 lma 0xc00ed000
Loading section .text.init, size 0x7c lma 0xc0306558
Loading section __ksymtab, size 0x4138 lma 0xc0307000
Loading section __ksymtab_gpl, size 0x1150 lma 0xc030b138
Loading section __kcrctab, size 0x209c lma 0xc030c288
Loading section __kcrctab_gpl, size 0x8a8 lma 0xc030e324
Loading section __ksymtab_strings, size 0xc040 lma 0xc030ebcc
Loading section __param, size 0x2e4 lma 0xc031ac0c
Loading section .data, size 0x1e76c lma 0xc031c000
Start address 0xc0008000, load size 3345456
Transfer rate: 64 KB/sec, 15632 bytes/write.
(gdb) cont
```

This will boot your kernel that was loaded into memory via JTAG.

Getting the kernel log buffer

Sometimes the kernel will panic before the serial is up and running. In such situations is it **very** handy to be able to dump the kernel log buffer. This can be done by looking at the content of the __log_buf in the kernel. In gdb this can be done by issuing

```
p (char*) &__log_buf[log_start]
```

There must be a simple way of printing the memory area between log_start and log_end.

The problem is that gdb stops after the first line. Currently we use this routine that copied from wchar.gdb until something "normal" came out. We defined dmesg it like this:

```
define dmesg
        set $__log_buf = $arg0
        set $log_start = $arg1
        set $log_end = $arg2
        set $x = $log_start
        echo "
        while ($x < $log_end)
                set $c = (char)(($__log_buf)[$x++])
                printf "%c" , $c
        end
        echo "\n
end
document dmesg
dmesg __log_buf log_start log_end
Print the content of the kernel message buffer
end
```

and call it like this:

```
dmesg __log_buf log_start log_end
```

Debugging a kernel module (.o and .ko)

Debugging a kernel module is harder.

Determining the module load address

gdb itself does not have knowledge about kernel modules and when debugging a kernel module. We will need to help gdb a little. One problem with modules is that it is not possible to determine where in the memory a module will be loaded before it is actually loaded so only once it is loaded we need to determine the address in memory it is loaded and tell gdb about it. There are many ways of determining this information. Here are 3 ways:

```
lsmod

cat /sys/module/mydriver/sections/.text

 #gdb implementation of the linux lsmod
 define lsmod
        set $current = modules.next
        set $offset =  ((int)&((struct module *)0).list)
    printf "Module\tAddress\n"

    while($current.next != modules.next)
                printf "%s\t%p\n",  \
                        ((struct module *) (((void *) ($current)) - $offset ) )->name ,\
                        ((struct module *) (((void *) ($current)) - $offset ) )->module_core
                set $current = $current.next
        end
end
```

Loading Symbols of a loaded module

```
add-symbol-file drivers/mydrivers/mydriver.o 0xbf098000
```

Note that we use the .o file and not the .ko one. The address at the end is the address where the kernel decided to load the module

Pointers

In the Linux Documentation directory under the kdump you will find file called gdbmacros.txt and it looks very promising as among other things it contains the a dmesg implementation

```
head  linux-2.6.22.1/Documentation/kdump/gdbmacros.txt
#
# This file contains a few gdb macros (user defined commands) to extract
# useful information from kernel crashdump (kdump) like stack traces of
# all the processes or a particular process and trapinfo.
#
```

A "find . -name "*gdb*" in the linux kernel directory also shows up a few interesting .gdbinit files that apparently can perform low level initialization.

Category:

- Development Tools

> From: eLinux.org

MagicSysRq

Note: This article is currently only a draft and is a part of a series of articles I'm going to publish the next few months - if you want to contribute to it, please feel free to. However it would be nice if you could coordinate your efforts with me

This article is still work in progress and still needs a lot of effort.

Contents

Introduction

MagicSysRq is a special key combination which lets the user perform certain low level tasks of the kernel and is especially useful when the whole system seems to be hung. MagicSysRq e.g. makes it possible to, more or less, cleanly shutdown a system, which doesn't even respond to CTRL+ALT+DEL any more.

The MagicSysRq functions are invoked by pressing ALT+SysRq+Function Key where function key describes one of the following.

TODO:Insert table here

Example

The probably most used combination of MagicSysRq functions is the one RSEIUB, which can be easyly remebered by the mnemonic trick ``Raising Elephants is so utterly boring\*footnote{``You can still find several references the* mnemonic *Raising Skinny Elephants is utterly boring* on the internet - however it is probably better to do the sync after the tasks exited/were killed} - although we're dealing with penguins here ;) According to the table above this performs, in order: R Put Keyboard in Raw Mode E terminate all tasks I kill all tasks S Sync U remount all filesystems read-only B Reboot

Troubleshooting

You can easily and safely check whether MagicSysRq works on your System by pressing Alt+SysRq+H - on a console you should now see a short help text of the MagicSysRq Feature or at least print this to `dmesg`.

If you don't see the help text, check whether CONFIG_MAGIC_SYSRQ=y is set in your Kernel Config. If it is set, please check the output of 'cat /proc/sys/kernel/sysrq' which should report 1 if MagicSysRq is enabled. To enable it just do a simple echo 1 > /proc/sys/kernel/sysrq

Important Files

/proc/sys/kernel/sysrq - enable/disable MagicSysRq at runtime /proc/sysrq-trigger - trigger MagicSysRq from the console (e.g. useful over ssh)

TODO Content:

- Explanation of remaining keys
- using it over serial line -> BREAK+key, in kermit ctrl-\ b KEY
- Interaction with KDB
- Interaction with kexec / crashkernel

External links

- Wikipedia Article about Magic_SysRq_key
- Kernel Documentation/sysrq.txt

Categories:

- Development Tools
- Tips and Tricks

External Links

- Embedded Linux JTAG debugging (CELF presentation)
- Debugging the Linux kernel using Eclipse/CDT and QEMU

Kernel Tracing and Profiling

About kernel tracing and profiling tools and their usage.

> From: eLinux.org

System Tap

This page has information about **System Tap**, which is of interest to embedded developers, because tracers are a useful tool for diagnosing problems during product development.

Contents

Introduction

SystemTap is a flexible and extensible system for adding trace collection and analysis to a running Linux kernel.

SystemTap is designed to be very flexible (allowing for the insertion of arbitrary C code), yet also easy-to-use (most trace statements are written in a simple scripting language, with useful data collection and aggregation routines available in (essentially) library form).

A key aspect of SystemTap is that it is intended to allow you to create a trace set (a "tapset"), and run it on a running Linux system, with no modification or re-compilation of the system required. To do this, it uses the kernel KProbes interface and loadable kernel modules to dynamically add probe points and newly generated code to the running kernel.

Open Source Projects/Mailing Lists

The main SystemTap site is at: http://sourceware.org/systemtap/

The SystemTap mail list archives are at: http://sourceware.org/ml/systemtap/

The tutorial, which gives a good overview of the system, is at: http://sourceware.org/systemtap/tutorial/

Miscellaneous notes

Probe types

There are several types of probes:

- kprobe & kretprobe, for dynamically insterted probes
- timers
- static instrumentation markers
- performance counter events

In the future, there may be:

- user-space probes,
- user-space return probes, and

- watchpoint probes (kernel & user)
- and more

See Also

Note that SystemTap is one of the major tracing systems for the Linux kernel.

There is work afoot (as of spring 2006) to try to collaborate on different parts of the tracing problem, between some of the major tracing projects. See the Tracing Collaboration Project page for more information.

ARM Support

System Tap works on ARM & OMAP platforms instructions are available here

Some Performance measurements

Jian Gui writes (in July 2006 on the **System Tap** mailing list):

```
Hi, we've tested the overhead of systemtap/LKET with some benchmarks
on a ppc64 machine.

It shows the overhead of systemtap/LKET is acceptable generally.
But it will also cause significant overhead for some benchmark of
special behavior, e.g. dbench. Dbench calls kill() in a very high
frequency to check whether a task is complete, thus leads to a high
overhead.

We categorized the event hooks into five groups in the testing:
grp1 - syscall.entry, process
grp2 - syscall.return, process
grp3 - iosyscall, ioscheduler, scsi, aio, process
grp4 - tskdispatch, pagefault, netdev, process
grp5 - syscall.entry, syscall.return, process

All the results are
   (score1 - score2)/score2 * 100%,  where:
score1: the benchmark score when probed by systemtap
score2: the benchmark score without probing

dbench (<3% is noise)
--------------------
grp1            -14.4%
grp2            -33.1%
grp3            -7.92%
grp4            -13.6%
grp5            -43.3%

specjbb (<3% is noise)
---------------------
grp 1           -0.87%
grp 2           -0.67%
grp 4           +0.47%
grp 5           +0.05%

tiobench (<3% is noise)
----------------------
grp1       sequential reads      +1.45%
           sequential writes     -6.98%
           random reads          +0.57%
           random writes         -2.11%
grp2       sequential reads      +0.11%
           sequential writes     -5.81%
           random reads          +0.03%
           random writes         -2.11%
grp3       sequential reads      +1.42%
```

```
          sequential writes     -6.98%
          random reads          +0.51%
          random writes         -2.11%
grp4      sequential reads      +1.38%
          sequential writes     -5.81%
          random reads          +0.60%
          random writes         -2.11%
grp5      sequential reads      +0.22%
          sequential writes     -8.14%
          random reads          -0.10%
          random writes         -1.05%

Rawiobench (<3% is noise)
------------------------
grp1      sequential aioread()     0%
          sequential aiowrite()    0%
          random aioread()         0%
          random aiowrite()        0%
grp2      sequential aioread()     0%
          sequential aiowrite()    0%
          random aioread()         0%
          random aiowrite()        -0.82%
grp3      sequential aioread()     0%
          sequential aiowrite()    0%
          random aioread()         0%
          random aiowrite()        0%
grp4      sequential aioread()     0%
          sequential aiowrite()    0%
          random aioread()         +0.79%
          random aiowrite()        -0.82%
grp5      sequential aioread()     0%
          sequential aiowrite()    -6.41%
          random aioread()         +0.79%
          random aiowrite()        0%

Test environment:
Machine:  Open Power 720/ 8 cpus/ 2 cores/ 6GB RAM (tiobench use 1G)
Software: RHEL4-U3GA/ 2.6.17.2/ systemtap-20060718/ elfutils-0.122-0.4
```

Links

- SystemTap Sans Kernel: A Pure Userspace Backend Slides Video

Category:

- Development Tools

> From: eLinux.org

Kernel Trace Systems

Here are some links to information about different kernel tracing systems:

Contents

General Purpose tracing systems

Some major Linux general-purpose tracing systems are:

- ptrace - ability to trace syscall entry and exit, and signal delivery, to a process (also used for debugging a process)
  - see "man ptrace" and "man strace"
- Ftrace
  - Ftrace Function Graph ARM - presentations and patches by Tim Bird to add function graph and duration tracing to ARM systems
    - The presentation has good introductory material on ftrace, as well as links to additional resources
  - tracer for kernel functions
  - can also be used for debugging or analyzing latencies and performance issues
  - in mainline since 2.6.27
  - See Measuring Function Duration with FTrace
    - outline of presentation by Tim Bird for Linux Symposium 2009
- System Tap - System Tap is a system for building and executing tracing and sampling systems that can be applied to a running Linux system
- LTTng - Linux Trace Toolkit, next generation
- LKST - Linux Kernel State Tracer

Special Purpose tracing systems

There are some other notable special-purpose kernel tracing systems:

- KFT - Kernel Function Trace - traces functions to show function durations and call graphs
- latency trace - RT-preempt tool for measuring interrupt and mutex latency
  - The latency tracer is embedded in the RT-preempt patch - see Realtime Preemption and RT-preempt Article
- block tracer (blktrace) - allows you to see exactly what is going on in the block layer for a given queue
  - Introduction by Jens Axboe: Introduction
  - Execellent presentation: blktrace.pdf
  - Guide to using is at: blktrace guide
  - This appears to have been mainlined as of 2.6.17
  - Timeline utility (blktrace post-processing tool): blktrace timeline utility
- delay accounting patches - collect statistics about the delays that are experienced by each task on the system
  - see delay accounting patches

Trace Infrastructure

- KProbes - grew out of dprobes, with information at: dprobes
  - see an excellent tutorial at: kprobes
  - The mainline version of the KProbes supports x86,Alpha and PPC64 architectures. A MIPS implementation has been completed on the 2.6.16 kernel and tested on the Toshiba TX49 platform. Patch is available in the Patch Archive.
- [would be nice to have some djprobe stuff here]

Sampling Systems

Note that profile systems (or "sampling systems") are slightly different, in that they involve sampling instead of event tracing. Some major ones for Linux are:

- top - provides a dynamic real-time view of a running system, including processes
  - see "man top"
  - see also ksysguard, Gnome system Monitor
- OProfile - system-wide profiler for Linux systems
  - see oprofile
  - also: oprofile at IBM
- BootChart - samples bootup and provides visualization of process startup and system utilization
  - see Bootchart

Related facilities

- in-kernel statistics infrastructure - proposal for a generic implementation of statistics facilities inside the kernel
  - see inkernel stats
- perfmon2 - interfaces to hardware performance monitoring features of the CPU
  - see perfmon
- inotify - inotify

Other Systems

Here are some systems I haven't classified yet:

- Datastreams - a system for creating and monitoring tracepoints - see datastreams

Collaboration Efforts

Some trace system project leaders are trying to collaborate: see Tracing Collaboration Project

Category:

- Linux tracing

> From: eLinux.org

Linux Trace Toolkit

@) ***Preliminary Draft*** @) under construction

Contents

Introduction

The Linux Trace Toolkit is used to examine the flow of execution (between processes, kernel threads, and interrupts) in a Linux system. This is useful for analyzing where delays occur in the system, and to see how processes interact (especially with regard to scheduling, interrupts, synchronization primitives, etc.)

⚠ * Note that as of 2.6.14, LTT is being replaced with LTTng. See LTTng

See also the announcement here: Patch: Linux Trace Toolkit Viewer/Next Generation announcement (LTTV/LTTng)

Rationale

Tracing is useful for analyzing stuff.

Downloads

LTTng

See the QUICKSTART guide at: Quickstart

Sample i386 trace for LTTng

Here is a zip file containing a sample trace on an i386 machine: i386-ltt-trace.zip

0.9.6 Release (Nov 21, 2004)

I found it a bit difficult to get the appropriate set of patches for LTT. After a bit of frustration, I built my own release for Nov. 23,

1. The release is 0.9.6 (no 'pre') and it works with the Linux kernel 2.6.9. The tar file for the release is in the PatchArchive.

Older stuff

The latest stable release is 0.9.5a, but this release is over 2 years old.

Here is the patch set I finally used, for kernel 2.6.8.1:

- message with ltt-2.6.8.1-relayfs.patch.bz2 by Mathieu Desnoyers
- message with ltt-2.6.8.1-ltt.patch.bz2 by Mathieu Desnoyers

getting attachments from ltt-dev archive

I had to use 'munpack' to extract the compressed patches from the mail messages in the archive. 'munpack' is part of the package 'mpack', which is available at: munpack

Patch

Here are some patches used by Tim Bird at Sony, targeted at MIPS, but known to also work on other platforms. This tarball contains both an all-in-one patch and a series file with discreet sub-patches.

-
    - media:Ltt-2.6.11-from-TimBird.tgz
- Patches for Linux kernel version 2.6.8.1, and !TraceToolkit tarfile are here:
    - ltt-2.6.8.1.tar.gz
    - TraceToolkit-0.9.6pre3-plus.tar.gz

These are bundles with multiple sub-patches and sub-tars. To install them, do the following:

- untar kernel patches:
    - `tar -xzvf ltt-2.6.8.1.tar.gz`
- apply patches:
    - `./tpm -t linux-2.6.8.1.tar.bz2 -f ltt-2.6.8.1.pl -o linux-2.6.8.1-ltt`
    - untar !TraceToolkit stuff
    - `tar -xzvf TraceToolkit-0.9.6pre3-plus.tar.gz`
- unpack and apply patch:
    - `./tpm -f TraceToolkit-0.9.6pre3-plus.pl -o TraceToolkit`

Now, follow the build and usage instructions for the software in the include docs (!TraceToolkit/Help/index.html). For cross-compiling, use the instructions on this wiki page.

Utility programs

How To Use

Building the software

Apply the ltt and relayfs patches to your kernel:

- configure kernel with LTT
- turn on RelayFS and LTT

make menuconfig `File systems ---> Pseudo filesystems --->`

- Relayfs file system support

(exit, exit) General Setup --->

- Linux Trace Toolkit support

You can leave klog debugging support turned off.

Compile the user-space tracedaemon program:

I couldn't figure out if the tracevisualizer tools handle cross-compilation correctly (by which I mean that you can natively compile the tracevisualizer but cross-compile the tracedaemon). Instead of blindly trying configure tricks, I instead used the instructions from Karim's book "Building Embedded Linux Systems". In a nutshell, the instructions go something like this:

Overview:

```
* install source for user-space stuff
* configure for native build
* hand-build tracedaemon specifying a cross-compiler
* build rest of user-space suite using native compiler
* install programs as appropriate
```

Detail (in my case):

```
* install !TraceToolkit-0.9.6pre3, apply "plus" patch
  ** tpm -f user-space.pl -o !TraceToolkit-0.9.6pre3
* cd !TraceToolkit-0.9.6pre3
* configure --prefix=/home/tbird/work/ltt/tools
* make -C !LibUserTrace CC=<cross-compiler-gcc> !UserTrace.o
* make -C !LibUserTrace CC=<cross-compiler-gcc> LDFLAGS="-static"
* make -C Daemon CC=<cross-compiler-gcc> LDFLAGS="-static"
* cp Daemon/tracedaemon Daemon/Scripts/trace Daemon/Scripts/tracecore
```

Daemon/Scripts/traceu \/usr/sbin

```
* make
* make install
```

Using the software

In April, 2004, Karim wrote: lkml thread) `The documentation is out of date. Basically, the createdev.sh script isn't needed anymore because of relayfs. You need to mount relayfs to use LTT. See the classic dox on filesystem mounting for this kind of thing. It's going to be something like: # mount -t relayfs nodev /mnt/relay`

There's no insmod for LTT. It isn't a device driver module, following LKML recommendations.

References

- LTT home page is at: LTT
- Presentation from OLS 2006:
  - LTTng Tracer : A low impact performance and behavior monitor for GNU/Linux
- Online documentation is at: LTT Documentation
  - This documentation is a bit old (2002), and has parts that are out of date.

Sample Results

Future Work

Here is a list of things that could be worked on for this feature:

- online reference needs to be updated
  - e.g. no mention of relayfs
- project seems fairly quiet, and unmaintained
- see if LTT patch should be refactored to take into account new kprobe support in the kernel

LKML Reaction to tracing

In Sep 2002, there was a thread about tracing, where some major kernel developers expressed their concerns about tracing infrastructure in the kernel.

- Ingo Molnar

```
my problem with this stuff is conceptual: it introduces a constant drag on the kernel sourcecode, while 99% of development will not want to trace, ever. When i do need tracing occasionally, then i take those 30 minutes to write up a tracer from pre-existing tracing patches, tailored to specific problems.</source> ... so use the power of the GPL-ed kernel and keep your patches separate, releasing them for specific stable kernel ranches (or even development kernels).
```

- Linus Torvalds

```
> To summarize: You find tracing useful, but software tracing is only of limited value in areas you're working at. What about other developers, which only want to develop a simple driver, without having to understand the whole kernel? Traces still work where printk() or kgdb don't work. I think it's reasonable to ask an user to enable tracing and reproduce the problem, which you can't reproduce yourself.
```

That makes adding source bloat ok? I've debugged some drivers with dprintk() style tracing, and it often makes the code harder to follow, even if it ends up being compiled away.

From what I've seen from the LTT thing, it's too heavy-weight to be good for many things (taking SMP-global locks for trace events is _not_ a good idea if the trace is for doing things like doing performance tracing, where a tracer that adds synchronization fundamentally _changes_ what is going on in ways that have nothing to do with timing).

I suspect we'll want to have some form of event tracing eventually, but I'm personally pretty convinced that it needs to be a per-CPU thing, and the core mechanism would need to be very lightweight. It's easier to build up complexity on top of a lightweight interface than it is to make a lightweight interface out of a heavy one.

Categories:

- Boot Time
- Tracing
- Kernel

> From: eLinux.org

LTTng

Linux Trace Toolkit Viewer/Next Generation

LTTV is a modular viewer/analysis tool specifically designed to deal with very large traces generated by a production system.

It comes with a Linux kernel tracer, Linux Trace Toolkit Next Generation (LTTng), which builds on the existing LTT tracepoints and RelayFS delivery mechanism but is a complete rewrite of LTT tracing module and daemon.

The design aims at facilitating contributions from the community. We know that the vast quantity of analysis that can be performed on trace data is practically unlimited, so we want to make it as easy and fun as possible to add those to the project.

You can get it at http://www.lttng.org. Follow the Quickstart guide to know how to get it from Debian packages, RPM or sources.

LTTV key features :

- Support for large traces (it has been tested with 15GB traces). I has been designed from the start to deal with huge traces.
- Information from several tracefiles can be combined in a single view on the fly.
- Deals with traces coming from any architecture size or endianness.
- Text command line interface supporting plugins for trace batch analysis.
- Graphical interface supporting visualisation plugins.
- Flexible event filter.
- Modular architecture :

```
- dynamically loadable plugins : each specific view/analysis becomes a plugin.
- module dependencies architecture for maximum functionality reuse and easier testing.
```

- Addition of new instrumentations (or any kind of trace point) becomes easier with an event description parser and a tracing code generator (genevent).

You can also get LTTng from http://www.lttng.org . Available in Debian, RPM and sources packages. See the Quickstart guide.

LTTng key features :

- Easy addition of new instrumentations by supporting the genevent code generator.
- Very precise timestamps on events by using the processor cycle counter as an unique monotonic time reference.
- Minimal impact on the traced system : instrumentation does not disable interrupts or take locks. It uses RCU and cmpxchg atomic operations instead.
- Non maskable interrupts (NMI) reentrancy.
- Supports many nestable types : structures, unions, arrays, sequences.
- Supports host dependent types.
- Makes dynamic alignment of trace data with a low overhead.
- Integration with LTTV viewer so tracing can be controlled directly from the graphical interface.
- Can record many (n) independent traces at once.
- Modular architecture.

http://lwn.net/Articles/166952/

External Links

- lwn.net: LTTng 2.0: Tracing for power users and developers - part 1
- lwn.net: LTTng 2.0: Tracing for power users and developers - part 2

Category:

- Linux

> From: eLinux.org

Ftrace

Ftrace is the Linux kernel internal tracer that was included in the Linux kernel in 2.6.27. Although Ftrace is named after the function tracer it also includes many more functionalities. But the function tracer is the part of Ftrace that makes it unique as you can trace almost any function in the kernel and with dynamic Ftrace, it has no overhead when not enabled.

The interface for Ftrace resides in the debugfs file system in the tracing directory. Documentation for this can be found in the Linux kernel source tree at Documentation/trace/ftrace.txt.

Contents

trace-cmd

Using the Ftrace debugfs interface can be awkward and time consuming. trace-cmd was created to interface with Ftrace using a binary tool which comes with full documentation in man pages.

Here's some examples of trace-cmd:

```
# trace-cmd record -e sched myprogram
```

The above will enable all the Ftrace tracepoints that are grouped under the sched system. You can find these tracepoints by looking at the debugfs system:

```
# mount -t debugfs nodev /sys/kernel/debug
# ls /sys/kernel/debug/tracing/events/sched
enable                  sched_process_fork   sched_stat_sleep
filter                  sched_process_free   sched_stat_wait
sched_kthread_stop      sched_process_wait   sched_switch
sched_kthread_stop_ret  sched_signal_send    sched_wait_task
sched_migrate_task      sched_stat_iowait    sched_wakeup
sched_process_exit      sched_stat_runtime   sched_wakeup_new
```

trace-cmd allows you to see the possible events without needing to look at this directory as well.

```
# trace-cmd list -e | grep sched:
sched:sched_kthread_stop
sched:sched_kthread_stop_ret
sched:sched_wait_task
sched:sched_wakeup
sched:sched_wakeup_new
sched:sched_switch
sched:sched_migrate_task
sched:sched_process_free
sched:sched_process_exit
sched:sched_process_wait
sched:sched_process_fork
sched:sched_signal_send
sched:sched_stat_wait
sched:sched_stat_runtime
sched:sched_stat_sleep
sched:sched_stat_iowait
```

You can find trace-cmd in its git repository.

Also within that same repository is KernelShark, which is a graphical user interface to trace-cmd. trace-cmd is built with just "make" and KernelShark is created with "make gui". This allows building trace-cmd on your embedded device and keeping the build from needing the GTK libraries required by KernelShark.

Tips

Tracing a specific process with the Ftrace interface

(Adapted from email by Steven Rostedt) To trace just the kernel functions executed in the context of a particular function, set the pseudo-variable 'set-ftrace-pid', to the process id (pid) of the process.

If the process is not already running, you can use a wrapper shell script and the 'exec' command, to execute a command as a known pid.

Like so:

```
#!/bin/sh
echo $$ > /debug/tracing/set_ftrace_pid
# can set other filtering here
echo function > /debug/tracing/current_tracer
exec $*
```

In this example, '$$' is the pid of the currently executing process (the shell script. This is set into the 'set_ftrace_pid' variable, then the 'function' tracer is enabled. Then this script exec's the command (specified by the first argument to the script).

Example usage (assuming script is called 'trace_command'):

```
trace_command ls
```

Tracing a specific process with trace-cmd

```
# trace-cmd record -p function -F ls
```

Capturing an oops (from startup) to the serial console

You can capture the function calls leading up to a panic by placing the following on the kernel command line:

```
ftrace=function ftrace_dump_on_oops
```

Note: You can also use 'ftrace=function_graph' if you would prefer that instead.

The ftrace documentation, in Documentation/trace/ftrace.txt mentions how to set ftrace_dump_on_oops in a running system, but I have found it very handy to have it configured to dump the trace from kernel startup, so that any panics that occur during boot (before user-space is started) are also captured.

Note that the output will be VERY long. Please be patient.

The output will look something like the following:

```
     ash-56      0d..2. 159400967us : _raw_spin_lock <-vprintk
     ash-56      0d..2. 159400972us : __raw_spin_lock <-_raw_spin_lock
     ash-56      0d..2. 159400974us : add_preempt_count <-__raw_spin_lock
     ash-56      0d..3. 159400978us : log_prefix <-vprintk
     ash-56      0d..3. 159400979us : emit_log_char <-vprintk
     ash-56      0d..3. 159400981us : console_trylock <-vprintk
     ash-56      0d..3. 159400983us : down_trylock <-console_trylock
     ash-56      0d..3. 159400985us : _raw_spin_lock_irqsave <-down_trylock
     ash-56      0d..3. 159400987us : __raw_spin_lock_irqsave <-_raw_spin_lock_irqsave
     ash-56      0d..3. 159400989us : add_preempt_count <-__raw_spin_lock_irqsave
     ash-56      0d..4. 159400991us : _raw_spin_unlock_irqrestore <-down_trylock
     ash-56      0d..4. 159400993us : sub_preempt_count <-_raw_spin_unlock_irqrestore
     ash-56      0d..3. 159400994us : _raw_spin_unlock <-vprintk
     ash-56      0d..3. 159400997us : sub_preempt_count <-_raw_spin_unlock
     ash-56      0d..2. 159400999us : console_unlock <-vprintk
     ash-56      0d..2. 159401000us : _raw_spin_lock_irqsave <-console_unlock
     ash-56      0d..2. 159401002us : __raw_spin_lock_irqsave <-_raw_spin_lock_irqsave
     ash-56      0d..2. 159401004us : add_preempt_count <-__raw_spin_lock_irqsave
     ash-56      0d..3. 159401006us : _raw_spin_unlock <-console_unlock
     ash-56      0d..3. 159401008us : sub_preempt_count <-_raw_spin_unlock
     ash-56      0d..2. 159401010us : _call_console_drivers <-console_unlock
     ash-56      0d..2. 159401012us : _call_console_drivers <-console_unlock
     ash-56      0d..2. 159401014us : _raw_spin_lock_irqsave <-console_unlock
     ash-56      0d..2. 159401015us : __raw_spin_lock_irqsave <-_raw_spin_lock_irqsave
     ash-56      0d..2. 159401017us : add_preempt_count <-__raw_spin_lock_irqsave
     ash-56      0d..3. 159401019us : up <-console_unlock
     ash-56      0d..3. 159401021us : _raw_spin_lock_irqsave <-up
     ash-56      0d..3. 159401023us : __raw_spin_lock_irqsave <-_raw_spin_lock_irqsave
     ash-56      0d..3. 159401024us : add_preempt_count <-__raw_spin_lock_irqsave
     ash-56      0d..4. 159401027us : _raw_spin_unlock_irqrestore <-up
     ash-56      0d..4. 159401029us : sub_preempt_count <-_raw_spin_unlock_irqrestore
     ash-56      0d..3. 159401031us : _raw_spin_unlock_irqrestore <-console_unlock
     ash-56      0d..3. 159401033us : sub_preempt_count <-_raw_spin_unlock_irqrestore
     ash-56      0d..2. 159401034us : wake_up_klogd <-console_unlock
     ash-56      0d..2. 159401037us : sub_preempt_count <-vprintk
     ash-56      0d..1. 159401039us : die <-__do_kernel_fault
     ash-56      0d..1. 159401041us : oops_enter <-die
---------------------------------
Modules linked in:
CPU: 0    Not tainted  (3.0.27_nl-kzm-a9-rt46-00022-g5e35327 #2)
PC is at sysrq_handle_crash+0x40/0x50
LR is at _raw_spin_unlock_irqrestore+0x34/0x54
pc : [<801b7dd4>]    lr : [<802f5420>]    psr: 60000093
sp : 9f0cdeb0  ip : 802f7ed4  fp : 9f0cdebc
r10: 9f0cdf68  r9 : 9fbaedc8  r8 : 00000000
r7 : 60000013  r6 : 00000063  r5 : 00000001  r4 : 8044b890
r3 : 00000000  r2 : 00000001  r1 : 20000093  r0 : 00000001
Flags: nZCv  IRQs off  FIQs on  Mode SVC_32  ISA ARM  Segment user
Control: 10c5787d  Table: 5f11c04a  DAC: 00000015
Process ash (pid: 56, stack limit = 0x9f0cc2f0)
Stack: (0x9f0cdeb0 to 0x9f0ce000)
dea0:                                     9f0cdee4 9f0cdec0 801b85d0 801b7da0
dec0: 9f0cdf68 00000002 801b867c 9f07a960 00000002 2ad20000 9f0cdefc 9f0cdee8
dee0: 801b86b0 801b852c 9f0cdf68 9fbaed80 9f0cdf2c 9f0cdf00 80108a84 801b8688
df00: 9f0cdf68 00000002 9f07a960 2ad20000 9f0cdf68 2ad20000 00000001 00000000
df20: 9f0cdf5c 9f0cdf30 800c2f14 80108a00 00000002 00000889 800c4978 00000002
df40: 9f07a960 0005a750 00000002 2ad20000 9f0cdfa4 9f0cdf60 800c3250 800c2e5c
```

```
df60: 00020001 0004423c 00000000 00000000 9f0cc000 00000000 9f0cdfa4 00000002
df80: 2ad20000 0005a750 00000004 8000db24 9f0cc000 00000000 00000000 9f0cdfa8
dfa0: 8000d8c0 800c3208 00000002 2ad20000 00000001 2ad20000 00000002 00000889
dfc0: 00000002 2ad20000 0005a750 00000004 00000001 00000000 2ad093b8 7e8766d4
dfe0: 00000000 7e8766c0 2ac20f08 2ac023ec 40000010 00000001 00000000 00000000
Backtrace:
[<801b7d94>] (sysrq_handle_crash+0x0/0x50) from [<801b85d0>] (__handle_sysrq+0xb0/0x15c)
[<801b8520>] (__handle_sysrq+0x0/0x15c) from [<801b86b0>] (write_sysrq_trigger+0x34/0x44)
 r8:2ad20000 r7:00000002 r6:9f07a960 r5:801b867c r4:00000002
r3:9f0cdf68
[<801b867c>] (write_sysrq_trigger+0x0/0x44) from [<80108a84>] (proc_reg_write+0x90/0xa4)
 r4:9fbaed80 r3:9f0cdf68
[<801089f4>] (proc_reg_write+0x0/0xa4) from [<800c2f14>] (vfs_write+0xc4/0x150)
[<800c2e50>] (vfs_write+0x0/0x150) from [<800c3250>] (sys_write+0x54/0x110)
 r8:2ad20000 r7:00000002 r6:0005a750 r5:9f07a960 r4:00000002
[<800c31fc>] (sys_write+0x0/0x110) from [<8000d8c0>] (ret_fast_syscall+0x0/0x30)
Code: 0a000000 e12fff33 e3a03000 e3a02001 (e5c32000)
---[ end trace feb441c6e3b9c3f1 ]---
Kernel panic - not syncing: Fatal exception
Backtrace:
[<80011908>] (dump_backtrace+0x0/0x114) from [<802f2304>] (dump_stack+0x20/0x24)
 r6:9f0cdd1c r5:8039bb64 r4:8045dc40 r3:00000002
[<802f22e4>] (dump_stack+0x0/0x24) from [<802f2404>] (panic+0xfc/0x220)
[<802f2308>] (panic+0x0/0x220) from [<80011dd4>] (die+0x18c/0x1d0)
 r3:00000001 r2:9f0cdd28 r1:20000113 r0:8039bb64
 r7:00000001
[<80011c48>] (die+0x0/0x1d0) from [<80014edc>] (__do_kernel_fault+0x74/0x94)
 r8:00000000 r7:9f0cde68 r6:9f0c1d40 r5:00000817 r4:00000000
[<80014e68>] (__do_kernel_fault+0x0/0x94) from [<802f7ca0>] (do_page_fault+0x254/0x274)
 r8:00000817 r7:9f0c1d40 r6:9f06d5e0 r5:00000000 r4:9f0cde68
r3:9f0cde68
[<802f7a4c>] (do_page_fault+0x0/0x274) from [<802f7db0>] (do_DataAbort+0x40/0xa8)
[<802f7d70>] (do_DataAbort+0x0/0xa8) from [<802f5d98>] (__dabt_svc+0x38/0x60)
 r8:00000000 r7:9f0cde9c r6:ffffffff r5:60000093 r4:801b7dd4
[<801b7d94>] (sysrq_handle_crash+0x0/0x50) from [<801b85d0>] (__handle_sysrq+0xb0/0x15c)
[<801b8520>] (__handle_sysrq+0x0/0x15c) from [<801b86b0>] (write_sysrq_trigger+0x34/0x44)
 r8:2ad20000 r7:00000002 r6:9f07a960 r5:801b867c r4:00000002
r3:9f0cdf68
[<801b867c>] (write_sysrq_trigger+0x0/0x44) from [<80108a84>] (proc_reg_write+0x90/0xa4)
 r4:9fbaed80 r3:9f0cdf68
[<801089f4>] (proc_reg_write+0x0/0xa4) from [<800c2f14>] (vfs_write+0xc4/0x150)
[<800c2e50>] (vfs_write+0x0/0x150) from [<800c3250>] (sys_write+0x54/0x110)
 r8:2ad20000 r7:00000002 r6:0005a750 r5:9f07a960 r4:00000002
[<800c31fc>] (sys_write+0x0/0x110) from [<8000d8c0>] (ret_fast_syscall+0x0/0x30)
CPU1: stopping
Backtrace:
[<80011908>] (dump_backtrace+0x0/0x114) from [<802f2304>] (dump_stack+0x20/0x24)
 r6:00000006 r5:00000001 r4:00000000 r3:00000000
[<802f22e4>] (dump_stack+0x0/0x24) from [<80008308>] (do_IPI+0xd8/0x148)
[<80008230>] (do_IPI+0x0/0x148) from [<802f5df4>] (__irq_svc+0x34/0xd0)
Exception stack(0x9fb47f68 to 0x9fb47fb0)
7f60:                   00000000 00000000 f300a000 00000000 9fb46000 80432444
7f80: 8045d464 802fd754 4000406a 411fc092 00000000 9fb47fbc 8000e660 9fb47fb0
7fa0: 8000e67c 8000e680 60000013 ffffffff
 r7:9fb47f9c r6:f0020000 r5:60000013 r4:8000e680
[<8000e64c>] (default_idle+0x0/0x38) from [<8000e920>] (cpu_idle+0x88/0xcc)
[<8000e898>] (cpu_idle+0x0/0xcc) from [<802f0130>] (secondary_start_kernel+0x140/0x164)
 r7:8045d57c r6:10c0387d r5:8043a2f8 r4:00000001
[<802efff0>] (secondary_start_kernel+0x0/0x164) from [<402efab4>] (0x402efab4)
 r5:00000015 r4:5fb4806a
```

Find latencies on kernel startup

It is possible to use ftrace to record functions that exceed a certain amount of time, using the 'tracing_thresh' option. This can be used for finding routines that are taking a long time on kernel startup, to help optimize bootup time:

- Make sure the following kerne configuration options are set:
  - CONFIG_FTRACE: "Tracers"
  - CONFIG_FUNCTION_TRACER: "Kernel Function Tracer"

    - CONFIG_FUNCTION_GRAPH_TRACER: "Kernel Function Graph Tracer"
- Use the following on the kernel command line:
    - tracing_thresh=200 ftrace=function_graph
        - this traces all functions taking longer than 200 microseconds (.2 ms). You can use any duration threshhold you want.
- to get the data:
    - $ mount -t debugfs debugfs /debug
    - $ cat /debug/tracing/trace

These command should be probably be done programatically (as part of an init script), to avoid data loss

- scope of operation
    - the tracer starts sometime during initialization, and you only get timings after it starts

Find deepest kernel stack

This is useful to find the routine with the deepest kernel stack The system continually monitors the stack depth of all processes, and whenever a low-water mark is hit (deepest stack), it records the list of functions.

(The following instructions work for a v3.0 Linux kernel)

- Kernel configuration: Set the following kernel configuration options:
    - CONFIG_FTRACE: "Tracers"
    - CONFIG_FUNCTION_TRACER: "Kernel Function Tracer"
    - CONFIG_STACK_TRACER: "Trace max stack"
- Turning it on: You can turn it on at boot time or at runtime.
    - At boot time, use the following on the kernel command line:
        - stacktrace
    - or, at runtime do:
        - echo 1 >/proc/sys/kernel/stack_tracer_enabled
- To get the data:
    - $ mount -t debugfs debugfs /debug
    - $ cat /debug/tracing/stack_trace
- scope of operation
    - the stack tracer will continue operating until you turn it off, which can be done with:
        - echo 0 >/proc/sys/kernel/stack_tracer_enabled

additional resources

See http://lwn.net/Articles/295955/

pytimechart

You can use pytimechart to explore ftrace traces visually. See http://packages.python.org/pytimechart/userguide.html

bootchart like traces with ftrace and pytimechart

You can use the following kernel command line parameters to generate a trace at boot, which can then be open with pytimechart to have a browsable bootchart.

```
trace_event=sched:*,timer:*,irq:* trace_buf_size=40M
```

output

Here is what the output looks like, on ARM:

```
/debug/tracing # cat stack_trace
        Depth    Size   Location    (42 entries)
        -----    ----   --------
  0)     3328      16   ftrace_test_stop_func+0x28/0x34
  1)     3312      28   __gnu_mcount_nc+0x58/0x60
  2)     3284      52   skb_release_data+0xc0/0xc8
  3)     3232      24   __kfree_skb+0x24/0xc0
  4)     3208      32   consume_skb+0xe4/0xf0
  5)     3176      56   smsc911x_hard_start_xmit+0x188/0x2f4
  6)     3120      72   dev_hard_start_xmit+0x440/0x6a4
  7)     3048      40   sch_direct_xmit+0x8c/0x1f8
  8)     3008      48   dev_queue_xmit+0x2c8/0x570
  9)     2960      56   neigh_resolve_output+0x32c/0x390
 10)     2904      40   ip_finish_output+0x2bc/0x37c
 11)     2864      32   ip_output+0xb0/0xb8
 12)     2832      24   ip_local_out+0x38/0x3c
 13)     2808      32   ip_send_skb+0x18/0xa4
 14)     2776      56   udp_send_skb+0x274/0x394
 15)     2720     240   udp_sendmsg+0x4dc/0x748
 16)     2480      32   inet_sendmsg+0x70/0x7c
 17)     2448     232   sock_sendmsg+0xa8/0x160
 18)     2216      32   kernel_sendmsg+0x40/0x48
 19)     2184      96   xs_send_kvec+0xa8/0xb0
 20)     2088      64   xs_sendpages+0x90/0x1f8
 21)     2024      40   xs_udp_send_request+0x4c/0x13c
 22)     1984      48   xprt_transmit+0x114/0x214
 23)     1936      40   call_transmit+0x208/0x27c
 24)     1896      48   __rpc_execute+0x88/0x334
 25)     1848      24   rpc_execute+0x68/0x70
 26)     1824      24   rpc_run_task+0xa8/0xb4
 27)     1800      64   rpc_call_sync+0x68/0x90
 28)     1736      32   nfs_rpc_wrapper.clone.6+0x3c/0x7c
 29)     1704      48   nfs_proc_getattr+0x70/0xac
 30)     1656      48   __nfs_revalidate_inode+0xe4/0x1f8
 31)     1608      56   nfs_lookup_revalidate+0x1ac/0x40c
 32)     1552      72   do_lookup+0x228/0x2e4
 33)     1480      72   do_last.clone.44+0x10c/0x688
 34)     1408      88   path_openat+0x2fc/0x394
 35)     1320     144   do_filp_open+0x40/0x8c
 36)     1176      40   open_exec+0x2c/0xc0
 37)     1136     136   load_elf_binary+0x1cc/0x12b8
 38)     1000      72   search_binary_handler+0x150/0x3a0
 39)      928      56   do_execve+0x170/0x328
 40)      872      32   sys_execve+0x44/0x64
 41)      840     840   ret_fast_syscall+0x0/0x30
```

Category:

- Linux tracing

> From: eLinux.org

Using Kernel Function Trace

Using Kernel Function Trace Ver. 0.1.1 -- 2007-04-26 Most material provided by Sony

---

Contents

Introduction

This document describes how to use Kernel Function Trace with the Linux kernel. It assumes you have already applied the KFT patch to your kernel, or that it was otherwise previously integrated with your kernel source code.

Kernel Function Trace (KFT) is a kernel function tracing system, which examines every function entry and exit in the Linux kernel. The KFT system provides for capturing a subset of these events along with timing and other details. KFT is different from other kernel tracing systems in that it is designed to be able to filter the events by the duration of the function calls. Thus, KFT is good for finding out where time is spent in functions and sub-routines in the kernel. When used in unfiltered mode, KFT is very useful to collect information about the flow of control in the kernel, which can help with debugging or optimizing kernel code.

The main mode of operation with KFT is by running a "dynamic" trace. That is, you start the kernel as usual, then, using the `/proc/kft` interface, configure a trace, start it, and retrieve the trace data immediately.

However, another special mode of operation is available for performing bootup time tracing. In this mode, the configuration for a trace is compiled statically into the kernel. This is sometimes referred to as "static" mode. This mode is useful for getting a trace of the kernel during system bootup, before user space is running and before any services are available to configure and start a trace. This mode is particularly helpful to find problems with kernel bootup time.

In either case, you specify a KFT configuration for the trace run. The configuration tells how to automatically start and stop the trace, whether to include interrupts as part of the trace, and whether to filter the event data by various criteria (for minimum function duration, only certain listed functions, etc.)

When a trace is complete, the event data collected during the trace is retrieved by reading from `/proc/kft_data`.

Finally, KFT provides tools to process and analyze the data in a KFT trace.

Quick Overview

Quick overview for using KFT in dynamic mode:

- Configure your kernel with support for KFT
- Compile your kernel
- Boot the kernel
- Write a configuration to `/proc/kft`
- Start the trace
- Read the trace data from `/proc/kft_data`
- Process the data
    - Use `addr2sym` to convert addresses to function names
    - use `kd` to analyze trace data

Quick overview for using KFT during bootup:

- Configure your kernel with support for KFT and KFT_STATIC_RUN
- Edit the configuration in `<linux_src>/kernel/kftstatic.conf`
- Compile your kernel
- Boot the kernel
    - The run should be triggered during bootup
- Read the trace data from `/proc/kft_data`
- Process the data
    - Use `addr2sym` to convert addresses to function names
    - use `kd` to analyze trace data

Detailed instructions

Configuring the kernel for using KFT

Configure your kernel to support KFT by editing the kernel configuration (.config) file.

For example, if you are using 'make menuconfig', set the following option under the "Kernel Hacking" menu.

```
Kernel Hacking --->
[*] Kernel Function Trace
```

Save this configuration. This will set the option CONFIG_KFT=y in your kernel .config file.

If you wish to perform a trace during kernel bootup time, also configure for KFT static mode.

For example, if you are using 'make menuconfig', set the following option under the "Kernel Hacking" menu.

```
Kernel Hacking --->
[*] Kernel Function Trace
[*] Static function tracing configuration
```

Save this configuration. This will set the following options in your kernel .config file:

```
CONFIG_KFT=y
CONFIG_KFT_STATIC_RUN=y
```

Editing the static trace run configuration (optional)

If you are performing a "static" trace, edit the file `kernel/kftstatic.conf` to set the configuration for the trace run you wish to perform at system boot. (see the next section "Configuring the trace run" for details on the trace configuration syntax and options.) Note that even if you perform or bootup time trace, you can still perform dynamic traces any time while the system is running.

Compiling the kernel

Build the kernel, and install it to boot on your target machine.

Make sure to save the `System.map` file from this build, since it will be used later when processing the trace data.

If you get an error compiling the kernel, see the next section on trouble-shooting configuration problems.

Configuring the trace run

To configure your trace, you write a trace configuration file. This file specifies when to start and stop the trace, and what events to save as part of the trace data.

Here is a sample configuration file, commonly used during bootup:

```
begin
  trigger start entry start_kernel
  trigger stop entry to_userspace
  filter mintime 500
end
```

This trace says to:

- start tracing when the function "start_kernel" is entered
- stop tracing when the function "to_userspace" is entered
- don't save the events for any function that takes less than 500 microseconds

The function "start_kernel" is the first C function executed by the kernel on startup. The function "to_userspace" is a function called immediately before execution is transferred to the first user space program (usually `/sbin/init`). This trace configuration says to start tracing immediately when the kernel starts executing, and stop tracing right before the first user space program runs. It will only save in the trace buffer a record of functions that took longer than 500 microseconds to execute.

Triggers

Triggers, in the configuration, are used to start and stop the data collection of the trace system. Triggers can be based on a function entry or exit event, or on the passage of time. The stop trigger is used to control the amount of data collected. The trace will automatically stop if the buffer runs out of space for trace data.

Time values are expressed in decimal microseconds. The start time is relative to booting, or to the initialization of the clock used for tracing (usually, whatever clock is being used by the internal kernel function sched_clock(). A stop time is relative to the start time.)

Here are some examples:

- statement: `trigger start entry start_kernel`
  - meaning: start tracing when the kernel enters the function "start_kernel"
- statement: `trigger stop exit do_fork`
  - meaning: stop tracing when the kernel exits the function "do_fork"
- statement: `trigger start time 10000000`
  - meaning: start tracing 10,000,000 microseconds (10 seconds) after booting
- statement: `trigger stop time 5000`
  - meaning: stop tracing 5,000 microseconds (5 milliseconds) after the trace starts

Filters

Filters control what data is collected during the trace. Since every kernel function entry and exit is a possible candidate for trace event recording, KFT can potentially generate a LOT of data. To control how much data is recorded, it is customary to set filters used during the trace.

You can filter by function duration, by interrupt context, or limit the trace to specific functions. Times in a filter statement are expressed in microseconds. Functions in a filter function list can be expressed by name in a static configuration, but must be expressed by addressed in a dynamic configuration.

Here are some examples:

- statement: `filter mintime 100`
  - meaning: only keep functions in the trace which last at least 100 microseconds
- statement: `filter maxtime 5000000`
  - meaning: discard functions in the trace which last more than 5,000,000 microseconds (5 seconds)
- statement: `filter noints`
  - meaning: discard functions in the trace which are executed when the processor is in interrupt context
- statement: `filter onlyints`
  - meaning: retain only the functions in the trace which are executed when the processor is in interrupt context
- statement: `filter funclist do_fork sys_read fend`
  - meaning: retain only events for the functions do_fork and sys_read.

Configuration scenarios

For other commands you can include in the trace configuration, see Appendix A

Handling a link error

You may get an error linking the kernel if you reference certain functions in the `kftstatic.conf` that are not visible globally. If you see a linker error like the following: "undefined reference to `foo_func'", then you can resolve this by making `'foo_func'` visible. Usually, this means finding the declaration of `'foo_func'`, and removing the `'static'` keyword from its declaration.

Initiating a KFT run

If you are running in static mode, upon booting the kernel, the trace should be initiated and run automatically, depending on the trigger and filter settings in `kernel/kftstatic.conf`).

If you are running in dynamic mode, then you initiate a run by writing a KFT configuration to `/proc/kft`, then "priming" the run.

Traces go through a state machine (a series of event transitions) in order to actually start collecting data. This is to allow trace collection to be separated from trace setup and preparation. The trace configuration specifies a start trigger, which will initiate the collection of data. When the configuration is written to `/proc/kft`, it is not ready to run yet. Making the trace ready to run is called "priming" it.

Therefore, the normal sequence of events for a trace run is:

1. . The user writes the configuration file, usually using an editor and creating the file in the local filesystem. Helper scripts can be used to auto-generate simple configurations for common tasks.
    i. . There is a helper script scripts/sym2addr, which converts function names in the configuration file to addresses. This can be copied to the target, along with the current System.map file, to make preparing the configuration file easier.
2. The user writes the configuration to KFT (via /proc/kft)
    i. e.g. `cat /tmp/trace.config >/proc/kft`
3. If needed, the user prepares for trace by setting up programs to run.
4. The user primes the trace
    i. e.g. `echo "prime" >/proc/kft`
5. A kernel event occurs which starts the trace (the start trigger fires)
6. Trace data is collected
7. A kernel event or buffer exhaustion stops the trace (that is, the stop trigger fires, or the buffer runs out)

It is possible to force the start or end of a trace using the `/proc/kft` interface. This overrides steps 5 or 7, which are normally performed by triggers in the trace configuration.

- To manually start a trace: `echo "start" >/proc/kft`
- To manually stop a trace: `echo "stop" >/proc/kft`

You can get the status of the current trace by reading `/proc/kft` .

To see the status of the currently configured trace:

- `cat /proc/kft`

Reading the trace data

When the trace is running, the trace data is accumulated in a buffer inside the kernel. Once the trace data is collected, you can retrieve if from the kernel by copying the data from `/proc/kft_data`. Usually, you will want to save the data to a file for later analysis.

Here is an example:

- `cat /proc/kft_data >/tmp/kft.log`

Processing the data

Copy the `kft.log` file from the target to your host development system (on which the kernel source resides), for example, into the `/tmp` directory on your host machine.

The raw `kft.log` file will only have numeric function addresses. To translate these addresses to symbols, use the `addr2sym` program, along with the `System.map` file which was produced when you built the kernel.

Change directory to your kernel source top-level directory and run `scripts/addr2sym` to translate addresses to symbols:

Example:

```
$ scripts/addr2sym /tmp/kft.log -m System.map > /tmp/kft.lst
```

Here is an example fragment of output from `addr2sym` on a TI OMAP Innovator. Entry and Delta value are times in microseconds. The Entry time is the time time since machine boot, and the Delta time is the time between the function entry and exit.

```
 Entry      Delta      PID            Function                    Called At
--------   --------   -----   -------------------------   -------------------------
   23662       1333       0                    con_init   console_init+0x78
   25375     209045       0             calibrate_delay   start_kernel+0xf0
  234425     106067       0                    mem_init   start_kernel+0x130
  234432     105278       0       free_all_bootmem_node   mem_init+0xc8
  234435     105270       0       free_all_bootmem_core   free_all_bootmem_node+0x28
  340498       4005       0       kmem_cache_sizes_init   start_kernel+0x134
```

In the above, `calibrate_delay` took about 209 milliseconds.

`mem_init` took 106 msecs, the majority of which (105 msecs) was in `free_all_bootmem_core` (which is called by `free_all_bootmem_node`, which is called by `mem_init`).

If you just look at the function duration, it may appear that lots of time is being spent in certain functions, when in reality those functions are "thin", and the real time-consuming function is one of its children. Thus, rather than look just at the function Delta (or duration), you should look at the function entry times. If there is a big leap in the function entry times, that means a lot of time was consumed in the function right before the leap.

In the example above, there is a leap from 234435 to 340498 (about 100 milliseconds) between the Entry times for `free_all_bootmem_core` and `kmem_cache_sizes_init`. No other functions Entries (lasting more than 500 microseconds, based on the KFT configuration used) were recorded during this time, so this means that this time was spent in `free_all_bootmem_core`.

CPU-yielding functions like schedule_timeout, switch_to, kernel_thread, etc. can have large Delta values due intervening scheduling activity, but these can often be quickly filtered out by following the "leaps in the entry times in the Entry column" above.

Analyzing the data with kd

You can use the program `"kd"` to further process the data. It is very helpful at this point to have resolved the names of the functions in the log file, but it is not strictly necessary. The `kd` program function reads a KFT log file and determines the time spent locally in a function versus the time spent in sub-routines. It sorts the functions by the total time spent in the

function, and can display various extra pieces of information about each function (number of times called, average call time, etc.)

Also `kd` can be used to re-generate a function call trace from the trace log. This can be very helpful to see the sequence of execution (including interrupts, context switches and returns) of the code that was traced.

Use "./kd -h" for more usage help.

As of this writing, KFT and kd do not correctly account for scheduling jumps. The time reported by KFT for function duration is just wall time from entry to exit.

For examples of what kd can show, try the following commands on the sample kft output file:

[show all functions sorted by time]

```
$ ./kd kftsample.lst | less
```

[show only 10 top time-consuming functions]

```
$ ./kd -n 10 kftsample.lst
```

[show only functions lasting longer than 100 milliseconds]

```
$ ./kd -t 100000 kftsample.lst
```

[show each function's most time-consuming child, and the number of times it was called. (You may want to make your terminal wider for this output.)]

```
$ ./kd -f Fcatlmn kftsample.lst
```

[show call traces]

```
$ ./kd -c kftsample.lst
```

[show call traces with timing data, and functions interlaced]

```
$ ./kd -c -l -i kftsample.lst
```

Note that the call trace mode may not produce accurate results if weird filtering was used in the trace config (routines that are part of the call tree may be missing, which will confuse kd).

KFT utilities

KFT includes the following helper scripts which are located in the kernel `scripts` directory:

- mkkftrun.pl - used during building the kernel to convert a configuration file into a C file to be compiled into the kernel. This is run automatically by the kernel make system. Users of KFT should not need to worry about this.
- sym2addr - convert function names to addresses in a KFT configuration file (for a dynamic trace). This is only used if a dynamic configuration has function names.
- addr2sym - convert function addresses to symbols in the trace data
- kd - KFT dump - does filtering, sorting, analysis and trace formatting of KFT trace logs

The use of most these are described elsewhere in this document. But this list is here for the sake of completeness.

[should provide usage for each command?]

Tips for using KFT

- How to look for long-duration functions?

```
(searching child functions for local time)
```

-
  - how to avoid being fooled by bogus local times
  - How to see a detailed function trace (don't use a min-filter)
  - How to interpret trace results including context switches

Online Resources

Here's a presentation about KFT usage: (Actually, the presentation covers KFT's predecessor KFI, but all the information is basically the same.)

- Learning the Kernel and Finding Performance Problems with KFI Presentation by Tim Bird at CELF International Technical Jamboree in 2005
  - Media:omap-serial_init.trace.txt
    - Sample trace used with presentation

Appendices

Appendix A - KFT configuration language

This appendix describes the language for specifying a KFT trace run. Is it used for both static mode (`kftstatic.conf`), and dynamic mode (written to `/proc/kft`).

A note on function names

NOTE that for parameters referencing functions, you can use the function name in `kftstatic.conf` (that is, when you using a static configuration). However, you have to use the function address when setting the configuration via the `/proc/kft` interface. The reason for this is that kernel symbols are always available at compile-time, but may not be available in the kernel at runtime, depending on your kernel configuration.

To convert a function name to an address, you can look up the address for the symbol in the System.map file for the current kernel. There is a helper program provided called `sym2addr` which you can use to convert the function names in a configuration file into addresses. To do this manually, use:

e.g. grep do_fork System.map

```
c001d804 T do_fork
```

In this case, you would put 0xc001d804 in place of the function name in the configuration file. (Note the leading '0x'.)

To use the helper function `sym2addr` , do the following:

```
sym2addr trace_do_fork.conf System.map >trace_do_fork.conf2
cat trace_do_fork.conf2 >/proc/kft
```

configuration block

The configuration for a single run is inside a block that starts with 'begin' and ends with 'end'. Inside the block are triggers, filters, and miscellaneous entries. By convention, each configuration entry is placed on its own line. When writing the configuration to /proc/kft, then the keyword "new" should appear before the block 'begin' keyword.

triggers

```
trigger:
         either "start" or "stop", and then one of:
        entry <funcname>
        exit <funcname>
        time <time-in-usecs>
syntax:
trigger start|stop entry|exit|time <arg>
```

Start time is relative to booting. Stop time is relative to trace start time.

filters

```
filters
    maxtime <max-time>
    mintime <min-time>
    noints
    onlyints
    funclist <func1> <func2> fend

syntax:
filter noints|onlyints|maxtime|mintime|funclist <args> fend
```

The funclist specifies a list of functions which will be traced. When a funclist is specified, only those functions are traced, and all other functions are ignored.

When specifying a configuration via /proc/kft, the 'fend' keyword must be used to indicated the end of the function list. When the configuration is specified via kftstatic.conf, no 'fend' keyword should be used.

watches

```
watches
    stack <low-water-threshold>
    worst-stack <starting-low-water-threshold>

syntax:
watch stack|worst-stack <threshold>
```

A watch is used to have KFT monitor the trace for a particular condition, and act on the condition (usually preserve extra data to help debug that condition). The only supported watches currently are for monitoring the stack depth.

For a "stack" watch, while the trace is running the current position of the stack pointer is checked upon entry to every function. If the stack position is lower than the specified threshold, the current call stack of functions is preserved in the log (no matter whether the functions match other KFT filtering criteria or not), and the function durations are marked with a -2 value, to highlight them in the log. This operation (saving the call stack) is performed every time the stack position underflows the threshold. In this mode, an arbitrary number of call stacks can be recorded in the log (up to the limit of the log size).

For a "worst-stack" watch, the same monitoring is performed as with a "stack" watch. However, every time the condition is met, the threshold (worst stack left) is set to the new low stack value. In this mode, a call stack is preserved for each new low-water condition. The last such set of marked functions in the log will record the most stack-consuming call stack seen during the trace. Note also that the lowest recorded stack position is available in the KFT status information (from /proc/kft).

miscellaneous

```
logentries <num-entries>
```

specify the maximum number entries for the log for this run

```
autorepeat
```

Repeat trace indefinitely. That is, on trace trigger stop, prime the trace to run again, but leave the data in the buffer. The trace will start again when the start trigger is matched, and stop again when the stop trigger is matched. The trace will stop autorepeating when the buffer becomes full.

```
# Other options that may be supported in the future:
# overwrite
# Overwrite old data in the trace buffer.  This converts the trace buffer to
# a circular buffer, and does not stop the trace when the buffer becomes full.
# In overwrite mode, the end of the trace is available if the buffer is
# not large enough to hold the entire trace.  In NOT overwrite mode (regular
# mode) the beginning of the trace is available if the buffer is not large
# enough to hold the entire trace.

# untimed
# Do not time function duration.  Normally, the log contains only function
# entry events, with the start time and duration of the function.  In
# untimed mode, the log contains entry AND exit events, with the start
# time for each event.  Calculation of function duration must be done by
# a log post-processing tool.

# prime
# Immediately prime the trace for execution.  "Priming" a trace means making
# it ready to run.  A trace loaded without the "prime" command will not be
# enabled until the user issues a separate "prime" command through the
# /proc interface.

# prime entry ??
# primt exit ??
# prime time ??
```

Configuration Samples

Here are some configuration samples:

Record all functions longer that 500 microseconds, during bootup. Don't include functions executed inside interrupts.

```
new
begin
  trigger start entry start_kernel
  trigger stop exit to_userspace
  filter mintime 500
  filter maxtime 0
  filter noints
end
```

Record all functions longer that 500 microseconds, for 5 seconds after the next fork don't worry about interrupts

Assuming 'do_fork' is at address 0xc001d804

```
new
begin
  trigger start entry 0xc001d804
  trigger stop time 5000000
  filter mintime 500
  filter maxtime 0
  filter noints
end
```

- record short routines called by do_fork
- use a small log

```
new
begin
  trigger start entry do_fork
  trigger stop exit do_fork
  filter mintime 10
  filter maxtime 400
  filter noints
  logentries 500
end
```

- record interrupts for 5 milliseconds, starting 5 seconds after booting

```
new
begin
  trigger start time 5000000
  trigger stop time 5000
  filter onlyints
end
```

- record all calls to schedule after 10 seconds
- Assuming schedule is at address

kftstatic.conf version:

```
begin
  trigger start time 10000000
  filter funclist schedule fend
end
```

/proc/kft version, assuming schedule is at c02cb754

```
new
begin
  trigger start time 10000000
  filter funclist 0xc02cb754 fend
end
```

Appendix B - Sample results

Here is an excerpt from a KFI log trace (processed with addr2sym). It shows all functions which lasted longer than 500 microseconds, from when the kernel entered start_kernel() to when it entered to_userspace().

kft log output (excerpt)

```
Kernel Instrumentation Run ID 0

Logging started at 6785045 usec by entry to function start_kernel
Logging stopped at 8423650 usec by entry to function to_userspace

Filters:
```

500 usecs minimum execution time

```
Filter Counters:

Execution time filter count = 896348
Total entries filtered = 896348
Entries not found = 24

Number of entries after filters = 1757



 Entry      Delta      PID            Function                    Called At
--------   --------   -----   ------------------------   ------------------------
       1          0       0              start_kernel   L6+0x0
      14       8687       0                setup_arch   start_kernel+0x35
      39        891       0              setup_memory   setup_arch+0x2a8
      53        872       0  register_bootmem_low_pages   setup_memory+0x8f
      54        871       0              free_bootmem   register_bootmem_low_pages+0x95
      54        871       0         free_bootmem_core   free_bootmem+0x34
     930       7432       0               paging_init   setup_arch+0x2af
     935       7427       0           zone_sizes_init   paging_init+0x4e
     935       7427       0            free_area_init   zone_sizes_init+0x83
     935       7427       0       free_area_init_node   free_area_init+0x4b
     935       3759       0       __alloc_bootmem_node   free_area_init_node+0xc5
     935       3759       0       __alloc_bootmem_core   __alloc_bootmem_node+0x43
    4694       3668       0       free_area_init_core   free_area_init_node+0x75
    4817       3535       0          memmap_init_zone   free_area_init_core+0x2bd
    8807     266911       0                 time_init   start_kernel+0xb6
    8807     261404       0             get_cmos_time   time_init+0x1c
  270211       5507       0              select_timer   time_init+0x41
  270211       5507       0                  init_tsc   select_timer+0x45
  270211       5507       0             calibrate_tsc   init_tsc+0x6c
  275718       1638       0              console_init   start_kernel+0xbb
  275718       1638       0                  con_init   console_init+0x59
  275954        733       0         vgacon_save_screen   con_init+0x288
  277376       6730       0                  mem_init   start_kernel+0xf8
  277376       1691       0          free_all_bootmem   mem_init+0x52
  277376       1691       0     free_all_bootmem_core   free_all_bootmem+0x24
  284118      25027       0           calibrate_delay   start_kernel+0x10f
  293860        770       0                   __delay   calibrate_delay+0x62
  293860        770       0                 delay_tsc   __delay+0x26
  294951       1534       0                   __delay   calibrate_delay+0x62
  294951       1534       0                 delay_tsc   __delay+0x26
  297134       1149       0                   __delay   calibrate_delay+0xbe
  297134       1149       0                 delay_tsc   __delay+0x26
 .
 .
 .
1638605          0     145             filemap_nopage   do_no_page+0xef
1638605          0     145                __lock_page   filemap_nopage+0x286
1638605          0     145                io_schedule   __lock_page+0x95
1638605          0     145                   schedule   io_schedule+0x24
1638605          0       5                   schedule   worker_thread+0x217
1638605          0       1               to_userspace   init+0xa6
```

The log is attached here: Media:Kfiboot-9.lst A Delta value of 0 usually means the exit from the routine was not seen.

kft log analysis with 'kd'

Below is a `kd` dump of the data from the above log.

For the purpose of finding areas of big time in the kernel, the functions with high "Local" time are important. For example, `delay_tsc()` is called 156 times, resulting in 619 milliseconds of duration. Other time-consuming routines were: `isapnp_isolate()` , `get_cmos_time()` , `default_idle()` .

The top line showing schedule() called 192 times and lasting over 5 seconds, is accounted wrong due to the switch in execution control inside the schedule routine. (The count of 192 calls is correct, but the duration is wrong.)

```
$ ~/work/kft/kft/kd -n 30 kftboot-9.lst
Function                 Count Time     Average  Local
------------------------ ----- -------- -------- --------
schedule                   192  5173790    26946  5173790
do_basic_setup               1  1159270  1159270       14
do_initcalls                 1  1159256  1159256      627
__delay                    156   619322     3970        0
delay_tsc                  156   619322     3970   619322
__const_udelay             146   608427     4167        0
probe_hwif                   8   553972    69246      126
do_probe                    31   553025    17839       68
ide_delay_50ms             103   552588     5364        0
isapnp_init                  1   383138   383138       18
isapnp_isolate               1   383120   383120   311629
ide_init                     1   339778   339778       22
probe_for_hwifs              1   339756   339756      103
ide_scan_pcibus              1   339653   339653       13
init_setup_piix              2   339640   169820        0
ide_scan_pcidev              2   339640   169820        0
piix_init_one                2   339640   169820        0
ide_setup_pci_device         2   339640   169820      242
probe_hwif_init              4   339398    84849       40
time_init                    1   266911   266911        0
get_cmos_time                1   261404   261404   261404
ide_generic_init             1   214614   214614        0
ideprobe_init                1   214614   214614        0
wait_for_completion          6   194573    32428        0
default_idle               183   192589     1052   192589
io_schedule                 18   171313     9517        0
__wait_on_buffer            14   150369    10740      141
i8042_init                   1   137210   137210      295
i8042_port_register          2   135318    67659      301
__serio_register_port        2   135017    67508        0
```

kft nested call trace with 'kd -c'

Below is a `kd -c` trace of the data from a log taken from a PPC440g platform, from a (dynamic) trace of the function do_fork().

Here is the configuration file that was used:

```
new
begin
  trigger start entry do_fork
  trigger stop exit do_fork
end
```

Here is the first part of the trace in nested call format: Times (Entry, Duration and Local) are in micro-seconds. Note the timer interrupt during the routine.

```
 Entry      Duration   Local       Pid    Trace
---------- ---------- ---------- ------- --------------------------------
         4      20428        209      33 do_fork
         7          6          6      33 |  alloc_pidmap
        18       2643         84      33 |  copy_process
        21        114         19      33 |  |  dup_task_struct
        24          8          6      33 |  |  |  prepare_to_copy
        27          2          2      33 |  |  |  |  sub_preempt_count
        35         22          9      33 |  |  |  kmem_cache_alloc
        38          2          2      33 |  |  |  |  __might_sleep
        43         11          9      33 |  |  |  |  cache_alloc_refill
        49          2          2      33 |  |  |  |  |  sub_preempt_count
        60         65          6      33 |  |  |  __get_free_pages
        63         59         14      33 |  |  |  |  __alloc_pages
        65          3          3      33 |  |  |  |  |  __might_sleep
        71          3          3      33 |  |  |  |  |  zone_watermark_ok
        77         37         17      33 |  |  |  |  |  buffered_rmqueue
        80          4          4      33 |  |  |  |  |  |  __rmqueue
        86          3          3      33 |  |  |  |  |  |  sub_preempt_count
        92          3          3      33 |  |  |  |  |  |  bad_range
        98          2          2      33 |  |  |  |  |  |  __mod_page_state
       103          8          5      33 |  |  |  |  |  |  prep_new_page
       106          3          3      33 |  |  |  |  |  |  |  set_page_refs
       117          2          2      33 |  |  |  |  |  zone_statistics
       141         25          4      33 |  |  do_posix_clock_monotonic_gettime
       143         21          6      33 |  |  |  do_posix_clock_monotonic_get
       146         15          6      33 |  |  |  |  do_posix_clock_monotonic_gettime_parts
       149          9          6      33 |  |  |  |  |  getnstimeofday
       152          3          3      33 |  |  |  |  |  |  do_gettimeofday
       169          3          3      33 |  |  copy_semundo
       174         41         17      33 |  |  copy_files
       177         19          9      33 |  |  |  kmem_cache_alloc
       180          2          2      33 |  |  |  |  __might_sleep
       185          8          5      33 |  |  |  |  cache_alloc_refill
       188          3          3      33 |  |  |  |  |  sub_preempt_count
       200          3          3      33 |  |  |  count_open_files
       209          2          2      33 |  |  |  sub_preempt_count
       218         19          8      33 |  |  kmem_cache_alloc
       220          2          2      33 |  |  |  __might_sleep
       225          9          6      33 |  |  |  cache_alloc_refill
       229          3          3      33 |  |  |  |  sub_preempt_count
       241          2          2      33 |  |  sub_preempt_count
       246        216          9      33 |  |  kmem_cache_alloc
       249        199        199      33 |  |  |  __might_sleep
----------- !!!! start --------------
       253        151         63      33 timer_interrupt
       256          8          6      -1 !  profile_tick
       259          2          2      -1 !  !  profile_hit
       267         61         15      -1 !  update_process_times
       270          8          5      -1 !  !  account_system_time
       273          3          3      -1 !  !  !  update_mem_hiwater
       281          8          5      -1 !  !  run_local_timers
       284          3          3      -1 !  !  !  raise_softirq
       293         27         16      -1 !  !  scheduler_tick
.
.
.
```

Appendix C - Using KFT for monitoring stack usage

```
* configure CONFIG_KFI_SAVE_SP (if saving the stack pointer as part of trace data)
*
```

Appendix D - Some notes on KFT operation

Trace overhead

KFT uses the "-finstrument-functions" capability of the gcc compiler to add instrumentation callouts to every kernel function entry and exit. This generates a large amount of overhead during kernel execution, even if a trace is not active. For this reason, KFT is turned off in the default configuration for your target board.

This high overhead means that using KFT may interfere with time-sensitive operations on your device. You should be careful when interpreting performance results on you device when KFT is configured on in your kernel, whether the results are obtained from KFT or from some other performance measurement tool. KFT is great at providing data for relative performance comparisons, but not for absolute performance timings.

Performance: KFT adds a fair amount of overhead to kernel execution. The reason for this is that the compiler adds instrumentation hooks to the start and end of every function. These hooks take additional time to execute. When a trace is active, even more time is used as events are compared against triggers and filters, and as events are logged to the trace buffer. It would be inappropriate to use an instrumented kernel for production use.

Local-time: Be careful when using the 'local time' numbers provided by 'kd'. These are calculated using the entry and exit times for the functions, and then subtracting the duration of other functions called during the top function's lifetime. However, due to filtering, interrupt handling, or context-switching, these numbers can be way off.

Overhead measurement

Mitsubishi measured the overhead of KFI (the predecessor to KFT) The period is from start_kernel() to smp_init().

Platform was: SH7751R 240MHz (Memory Clock 80MHz)

```
With KFI    : 922.419 msec
Without KFI : 666.982 msec
Overhead    : 27.69%
```

Trace buffer exhaustion

Because every function in the kernel is traced, with certain trace configuration settings it is possible to VERY rapidly fill up the trace buffer. Kernel functions are executed several thousand times a second, even when the machine appears to be doing nothing.

The trace buffer is not circular. As soon as the buffer fills up with data, the trace capture automatically stops. For this reason, it is common to have the trace buffer exhausted during a trace.

How to fix:

```
* increase the trace buffer size
* use filters
  * filter only by certain functions
  * increase the minimum function duration to save in the trace
```

Q. Is there a way to adjust the trigger or filters to reduce the memory usage?

A. The memory usage is determined by the size of the log, which is specified by `logentries` in the KFT configuration. If `logentries` is not specified, it defaults to a rather large number (20,000 in the current code). To use a smaller trace log, specify a smaller number of logentries in the KFT configuration.

The use of triggers and filters can help you fit more data (or more pertinent data) into the log, so you can more readily see the information you are interested in.

By setting start and stop triggers with a narrower "range" of operation, then the amount of data put into the log will be more limited. For example, the default configuration for a static trace uses

```
trigger start entry start_kernel
trigger stop entry to_userspace
```

This will trace EVERYTHING that the kernel does between those two routines. However, you can limit tracing to a much smaller time area of kernel initialization using better triggers. Here is an example showing a triggers for just watching mem_init():

```
trigger start entry mem_init
trigger stop exit mem_init
```

Filters are also vital to reduce the number of entries the trace log. With no time filters in place, KFT will log every single function executed by the kernel. This will quickly overrun the log (no matter what size you have reserved with `logentries`.

When using KFT to find long-duration functions in the kernel, we usually are not interested in routines that execute quickly, and instead use something like "filter mintime 500" to filter out routines taking less than 500 microseconds.

Early clock issues

On many platforms, the clock used for performing trace timings is not available immediately when the kernel begins execution. Often, the clock is initialized sometime during the time_init() function of kernel startup. In this case, the function entry times and durations may be incorrect, for functions which begin before the clock is set up. Also time-related filters will not operate correctly on these functions. Usually, this is not a problem, since the times come back as zero, and any minimum time filters in place will remove the events from the trace buffer.

The result of all this is that, on machines where the clock is not immediately available at kernel start, there will be a "blind spot" during initialization, which is effectively not traceable by KFT. You can get event data for this "blind" period, by turning off the time filter for events, but this will result in a very large set of events (all without valid timing information) at the beginning of the event log.

Adding platform support for the KFT clock source

By default, KFT uses sched_clock() as the clock source for event timings. This is called from the routine kft_readclock().

sched_clock() is new in the 2.6 kernel, and returns a 64-bit value containing nanoseconds (not necessarily relative to any particular time base, but assumed to be monotonically increasing, and relatively frequency-stable.)

If your platform has good support for sched_clock(), then KFT should work for you unmodified. If not, you may wish to do one of two things:

- improve support for sched_clock() in your board port, or
- write a custom kft_readclock() routine.

A "good" sched_clock() routine will provide at least microsecond resolution on return values. Some architectures have sched_clock() returning values based on the `jiffy` variable, which on many embedded platforms only has resolution to 10 milliseconds.

There are some sample custom kft_readclock() routines in the current code for different architectures. These alternate routines are not active, via pre-processor conditionals. However, you can use them for samples of how to write your own custom KFT clock routine.

Modification History

Date	Version	Description	
2006-10-13	0.1.0	First draft of document	
2007-04-26	0.1.1	Fixed "run run" typo, and added some material on filters and triggers	

Categories:

- Development Tools
- Tips and Tricks

> From: eLinux.org

Linux Kernel State Tracer

Table Of Contents:

Contents

Description

[This section describes the technology ....]

LKST is Kernel State Tracer for Linux in order to keep track of Kernel Event such as:

```
- Process Management
- Interrupt
- Exceptions
- System Calls
- Memory Managements
- Networking: sending packets, receiving packets
- Sys V IPC
- Locks
- Timer
- Oops
```

Originally LKST is developed for Linux Enterprise Systems and now we have port it to Reference Boards for Embedded Systems and currently SH4 port(RTS7751R2D), MIPS/TX49 port (RBHMA4400CE) and ARM/OMAP port (TI OMAP INNOVATOR/OSK) are available.

Rationale

LKST is one of a number of tracing systems available for the Linux kernel. Such event tracing systems are very useful for analyzing kernel behaviour, and learning how interrupts, kernel threads and user-space applications interact on the system.

Resources

Projects

Here is some information about LKST:

- project home page: http://lkst.sourceforge.net/

Documents and presentations

- Media:CELF_LKST_SH_Presen-2005-1.pdf
  - presentation given by Hitachi at CELF Jan 2005 technical conference.
- Media:CELF_LKST_SH_Lineo-2005-2.pdf
  - presentation given by Lineo at CELF Jan 2005 technical conference.
- Media:HITACHI-LKST-CELF-200601.pdf
  - presentation given at International Technical Conference, June 2006
- Media:CELFTokyoJam6_LkstUpdate_Lineo.pdf
  - presentation 'Features of lkslogtools' given at CELF Jan 2006 technical jamboree (6)

Specifications

Downloads

Patch

- You can acquire patches at: http://sourceforge.net/projects/lkst/
- and click the link of |Patches|: http://sourceforge.net/tracker/?group_id=41854&atid=431465

Utility programs

[other programs, user-space, test, etc. related to this technology]

How To Use

How to validate

[put references to test plans, scripts, methods, etc. here]

Sample Results

[Examples of use with measurement of the effects.]

Case Study 1

Case Study 2

Status

- Status: [implemented]

```
(one of: not started, researched, implemented, measured, documented, accepted)
```

- Architecture Support:

```
(for each arch, one of: unknown, patches apply, compiles, runs, works, accepted)
```

-
    - i386: works
    - x86_64: works
    - ia64: works
    - ARM: runs
    - PPC: unknown
    - MIPS: runs
    - SH: runs

Future Work/Action Items

Here is a list of things that could be worked on for this feature:

Category:

- Development Tools

Android Portal

Android is a software platform and operating system written by Google and the Open Handset Alliance, designed for use in small form factor devices and smartphones.

Getting Started

Overview of Android system, from versions, history, design and architecture to the tools and tutorials.

> From: eLinux.org

Android Intro

Contents

Overview

Android is a software platform and operating system written by Google and the Open Handset Alliance, designed for use in small form factor devices and smartphones.

The Android wikipedia entry is a really good place to get an overview of Android capabilities, history and direction. I won't duplicate the information from that site here.

The official web site for Android is http://www.android.com/. From here you can find links to:

- Android Market - This is the place where developers can post (for free or for sale) applications to run on Android-based devices, and where users can download these applications.
- Android Developer Site - Where software developers can learn how to write applications for Android devices, or modify aspects of the Android software itself.
- Android Open Source Project - This is where you can find the source code for the Android operating system (including Linux kernel, the libraries, Dalvik VM and rest of the Android system.)
- The Growth of Android in Embedded Systems
  - This Linux Training publication has some excellent material on the current (as of early 2013) status of Android as it relates to traditional embedded linux.

Clicking on the "What is Android" tab takes you to: http://www.android.com/about/, which has some high-level bullet points, and a video with a little bit of Android history.

Videos

Prototype Video

In November, 2007, Google released a video showing some of the features of their early prototype work on Android. See http://www.youtube.com/watch?v=1FJHYqE0RDg

G1 Product Video

The first phone product based on Android was the G1, manufactured by HTC and shipped by T-Mobile.

The G1 home page is at: http://www.t-mobileg1.com/. This site contains many videos and voiceovers describing G1 and Android features.

Engadget Hand's-on walkthrough of G1 features (September, 2008) http://www.engadget.com/2008/09/23/video-android-walkthrough-on-t-mobile-g1/

Technical Information

A good place to start, for technical information, is Google's "What is Android?" page

Another good place with technical information is Android FAQ

Also, go to Android Architecture on this site.

Category:

- Android

> From: eLinux.org

Android Architecture

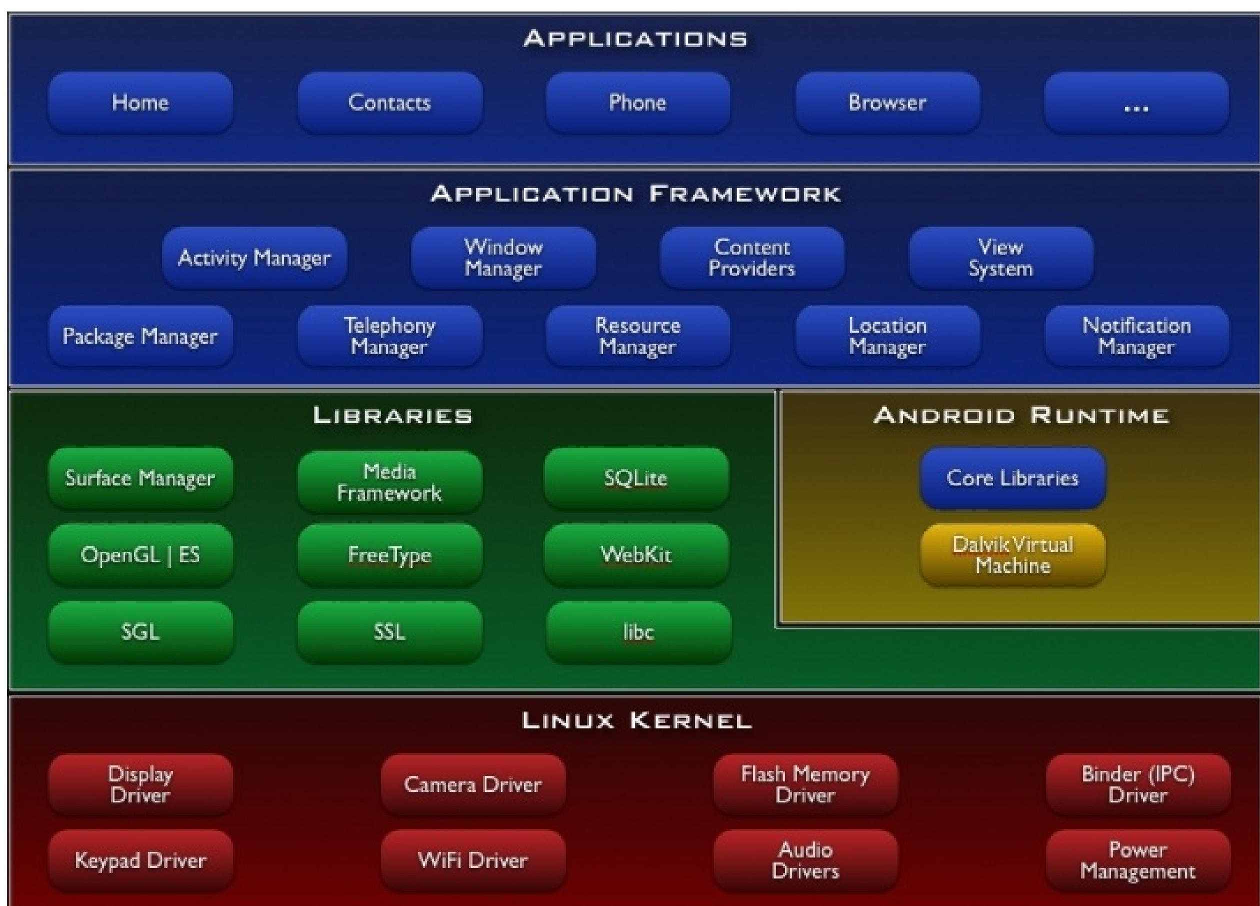
See Google's What is Android? page for an overview of Android components, and a diagram of the architecture.

The diagram on that page appears in every presentation I have ever seen about Android technical topics (with the exception of my own).

Contents

Architecture Diagram

Here is the Android Architecture Diagram, obtained from here.



See also Android internals diagram

Basically Android has the following layers:

- applications (written in java, executing in Dalvik)

- framework services and libraries (written mostly in java)
  - applications and most framework code executes in a virtual machine
- native libraries, daemons and services (written in C or C++)
- the Linux kernel, which includes
  - drivers for hardware, networking, file system access and inter-process-communication

Overview presentations

- Android is not just Java on Linux
  - Great presentation by Tetsuyuki Kobayashi overview of Android
- See this Android Internals presentation by Karim Yaghmour
  - http://www.opersys.com/blog/android-internals-101103
  - You'll find both the video and the slides there
- Mythbusters_Android.pdf Presentation by Matt Porter at ELC Europe
  - Has bits and pieces showing problematic Android code and policies

Breakdown of running Android system

A quick look at Android contents and programs running when Android starts is at:

- http://benno.id.au/blog/2007/11/13/android-under-the-hood

Relation to the Linux kernel

Here is Greg Kroah-Hartmans presentation on Android from the CELF conference 2010, discussing how Google/Android work (or don't work) with the Linux community.

Java

Java is used as a language for application programming, but it is converted into a non-java byte code for runtime interpretation by a custom interpreter (Dalvik).

Java/Object Oriented Philosophy

Practicality is more important than purity in implementing the Android system.

Dianne Hackborn, one of the principal engineers working on Android, wrote:

It's not like I am a C programmer who doesn't like object oriented design. In fact prior to Android my primary language was C++... and honestly, Java really annoys me in the way it introduces so much more overhead to do things that I could express in very nice OO concepts in C++ with a much lighter-weight result.

Though Java has a lot of other nice attributes that make it good for Android, it also has its share of design flaws and misfeatures that mean we can't be totally beautifully OO as you would like.

Finally, going forward, our API conventions were defined in a way that allowed us to ship a well performing system on the hardware we had the time. As the situation changes (and it slowly is, but not enough yet) that could change... however, I will probably lean towards keeping those API conventions in place just for the sake of consistency with everything that currently exists. Of course if Android is successful and in 10 years from now we are designing a whole new next generation Android framework... well, then the world is a different place.

Category:

- Android

> From: eLinux.org

Android Tools

Here are some development tools useful for working with Android

Contents

Android SDK

host-side tools

adb

adb is the android debugger - it also doubles as file transfer agent. The setup consists of an `adbd` on the target in the `/sbin` directory. On the host two programs are run: the `adb` application (in the SDK's `tools` directory) and an adb server, started by the adb application.

For emulators, adb will usually run automagically.

For real boards - with debugging over USB, you might need to do work, as is documented here: http://developer.android.com/guide/developing/device.html#setting-up .

For real boards that do not have a USB connection but have Ethernet instead, you might need to do a few tricks.

- make sure that adbd runs on the board. If it doesn't run, you might want to check the `init.rc` file.
- make sure that the network connection between host and the board is working - test pinging both ways.
- on the host, type the following (you need to specify the board's IP address on the host):

```
ADBHOST=<target-ip> tools/adb kill-server
ADBHOST=<target-ip> tools/adb shell
```

- you should now get a prompt on the board, you can exit the prompt if you want.
- `tools/adb devices` should now list the device.

Running adbd on non-Android systems

It is sometimes useful to use adbd on non-Android embedded Linux systems. Here is a patch that can be applied to adb (source as of 2014-04-05) to change it to avoid "Android-isms" in the build. Instructions are in a README.NONANDROID.TXT file.

File:0001-Add-support-for-non-Android-use-of-adbd.patch

This patch can be applied to your adb source by cd'ing to the directory /system/core/adb, applying the patch with:

```
$ git am 0001-Add-support-for-non-Android-use-of-adbd.patch
```

aapt

The Android Asset Packaging Tool is used to create, inspect and manage Android packages.

You can use this to see details about a package, it's resources, and xml information.

The Android developer page on aapt is somewhat meager.

See Android aapt for substantially more information.

ddms

The Dalvik Debug Monitor Server is a host-based tool which interacts with and Android target system and can show numerous bits of information, including the log, cpu and memory utilization, and lots of details about individual processes.

See the DDMS developer guide

Fastboot

Android Fastboot is a tool to boot and manipulate the partitions on an Android development phone.

Toolchains

Android provides pre-built toolchains (C/C++ compilers and linkers), but requires the installation of a java compiler (JDK) from an external source.

As of NDK version r5 (December 2010), the toolchains can now be used in standalone cross-compiler mode. See docs/STANDALONE-TOOLCHAIN.html in the NDK for information about this. Previously, the toolchains could be used within the build system, but it was difficult and error prone to compile native programs outside the Android build system with them.

Emulator

- Emulator - See http://developer.android.com/guide/developing/tools/emulator.html

The emulator is a version of QEMU, which mimics the instruction set of an ARM processor, and the hardware that one might find on a mobile phone. The emulator runs on an x86 system, but executes an ARM linux kernel and programs. The flow of control is:

-
    - application ->
    - dalvik VM ->
    - C/C++ libraries ->
    - ARM linux kernel ->
    - emulated instructions and hardware (QEMU)->
    - C libraries->
    - x86 kernel ->
    - real hardware

traceview

- Google's main page describing traceview: http://developer.android.com/guide/developing/tools/traceview.html
- http://www.bottomlesspit.org/file_download/2/Android_SDK_Traceview_tool.pdf
    - good overview presentation by Olivier Bilodeau
    - presentation with speaker notes: http://www.bottomlesspit.org/file_download/3/Android_SDK_Traceview_tool_w_speakernotes.pdf
- Performance Tuning Android Applications
    - straightforward article discussing traceview use to find an application bottleneck. April 2009.

target-side tools

am

Activity Manager - can be used to start applications at the command line, or send intents to running applications.

dumpstate

dumps the state of the system. It scans the /proc filesystem, and collects various system properties, and puts them in a single report, suitable for sending to someone for support or development help.

logcat

This is the user tool for accessing the Android system log. This is implemented at a special option in adb (I'm not sure what the difference is between "adb logcat" and "adb shell logcat")

You can find lots of information about logcat on the Android logger page, and at http://developer.android.com/guide/developing/tools/adb.html#logcat

monkey

procrank

procrank shows a listing of processes on the system, sorted by one of the memory utilization metrics. See Android Memory Usage#procrank

service

Can be used to send an individual service message.

```
Usage: service [-h|-?]
       service list
       service check SERVICE
       service call SERVICE CODE [i32 INT | s16 STR] ...
Options:
   i32: Write the integer INT into the send parcel.
   s16: Write the UTF-16 string STR into the send parcel.
```

On one forum, I saw that you could switch between portrait and landscape with:

```
$ service call window 18 i32 1 # to set to landscape on the emulator
$ service call window 18 i32 0 # to set to portrait on the emulator
```

service list will show a list of different services that can be communicated with.

sqlite3

sqlite3 is a command-line database client program, for manipulating sqlite databases.

See http://www.higherpass.com/Android/Tutorials/Accessing-Data-With-Android-Cursors/ for a tutorial and some examples of using sqlite.

toolbox

toolbox is the equivalent of busybox on an Android system. That is, it is a multi-function program that provides many difference commands from a single binary. this includes things like: ps, ls, top, stop, start - commands to stop and start services on an Android system

See Android toolbox for details about individual commands.

other tools

agcc

- agcc - A wrapper tool for compiling native Android apps (linked directly to bionic)
  - See http://android-tricks.blogspot.com/2009/02/hello-world-c-program-on-using-android.html

bootchart

- See Using Bootchart on Android

busybox

Android ships with a utility suite (called 'toolbox') that is not busybox.

You can get a binary busybox for Android here The site includes instructions for easy installation on your device.

If you're interested in including busybox into a platform build:

- precompiled binaries here
- a presentation on how to build (or not) and integrate busybox into platform build (slides available here).

smem

- smem - smem is a tools for analyzing the memory usage on a system
  - See Using smem on Android for more information

strace

- strace
  - Statically linked binary available at: http://benno.id.au/blog/2007/11/18/android-runtime-strace
  - Instructions for building Android strace - http://discuz-android.blogspot.com/2008/01/create-google-android-strace-tool.html

Eclipse

The officially supported integrated development environment (IDE) is Eclipse (currently 3.4 or 3.5) using the Android Development Tools (ADT) Plugin, though developers may use any text editor to edit Java and XML files then use command line tools (Java Development Kit and Apache Ant are required) to create, build and debug Android applications as well as control attached Android devices (e.g., triggering a reboot, installing software package(s) remotely).

Hardware

Serial Cable for G1

You can build a serial cable to use with the G1, which is helpful to see kernel boot messages on the serial console.

See http://www.instructables.com/id/Android_G1_Serial_Cable

Back to Android Portal

Category:

- Android

> From: eLinux.org

Android Glossary

Here are some Android terms (some even with definitions!!)

See also the developer glossary at: http://developer.android.com/guide/appendix/glossary.html

Back to Android Portal

Contents

A

aapt Android Asset Packaging Tool - a tool for creating, inspecting and modifying Android application packages.

Activity A single focused thing the user can do on an Android device. Also, a java class in the Android framework, which is used as the superclass for an Activity implementation. See http://developer.android.com/reference/android/app/Activity.html. "An Activity presents a visual user interface for one focused endeavor the user can undertake." See Activity on the page: http://developer.android.com/guide/topics/fundamentals.html

adb Android Debug Bridge - a tool for communicating between the host and a target Android system (including an emulator running on the host). See http://developer.android.com/guide/developing/tools/adb.html

ADP1 Android Developer Phone 1

ANR Application Not Responding - this is a type of bug where Android believes a process has hung. The system may kill the process, and leave information about it in /data/anr for post-mortem analysis.

Android A robot resembling a human being - the name of the operating system produced by Google for mobile phones. Apparently, Andy Rubin, one of the original founders of Android, Inc. loves robots.

AndroidManifest.xml A file describing the contents, permissions and other attributes of an Android application package. See http://developer.android.com/guide/topics/manifest/manifest-intro.html

Android, Inc. A company founded by Andy Rubin and others to create a mobile phone operating system. Android, Inc. was acquired by Google in 2005.

ASE Android Scripting Environment - the old name of Scripting Layer for Android. See Android Scripting

B

Binder An Interprocess Communication (IPC) mechanism. See http://cs736-android.pbworks.com/IPC-Binder and http://groups.google.com/group/android-developers/msg/dc0e0e872de9b0d2

Bionic small C library used in Android devices

Bootchart A mechanism to create visual charts of a Linux boot sequence, including the timing of process start and execution. See Using Bootchart on Android

C

Cliq The US name for the Motorola Android phone.

Content Provider An piece of software on an Android system that provides information (content) to other software elements. See http://developer.android.com/guide/topics/providers/content-providers.html. Also, a class which is the superclass for code which acts as a content provider. See the ContentProvider class documentation

Cupcake The code name for Android version 1.5.

D

Dalvik Virtual Machine in which Android applications are run. This VM executes Dalvik bytecode, which is compiled from programs written in the Java language. Note that the Dalvik VM is not a Java VM (JVM).

Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool.

See Android Dalvik VM for more information

Donut The code name for Android version 1.6

Dream Code name for the mobile phone hardware publicly called the t-Mobile G1, in the United States.

Droid The name for an upcoming Android phone by Motorola (I believe this is the high-end phone, and Cliq is the low-end phone?)

E

Eclair The code name for Android version 2.1

F

fastboot a program which communicates with the developer firmware, and which is capable of loading new software on the ADP1 phone (including re-writing the flash partitions on the device). See Android Fastboot

FreeType An open-source set of fonts and font system

Froyo *Frozen Yogurt* - The code name for Android version 2.2

G

G1 The name of the first Android-based mobile phone, from t-Mobile.

Galaxy The name of the first Samsung Android phone

Gingerbread The code name for Android version 2.3

Goldfish The name of a virtual ARM platform provided by the emulator.

Goldfish executes ARM926T instructions and has hooks for input and output -- such as reading key presses from or displaying video output in the emulator. There is a "goldfish" configuration file for compiling the Linux kernel to run with this emulated platform.

Google A large web search company, and primary developer of Android

H

Honeycomb The code name for Android version 3.0 - especially targeted at table computers

I

Ice Cream Sandwhich Android version 2.4 or 3.1 - the successor to Gingerbread and/or Honeycomb (possibly indicating a development fork) See http://techcrunch.com/2011/01/11/android-ice-cream-sandwich/

init the first user-space program run in the Android system. It is not a standard Linux-style 'init' program (which processes an /etc/inittab file). Rather, it processes a script called init.rc in the root directory of the file system. See Android Booting#'init'

Intent A facility to send messages between different Android components. A message is conveyed using an Intent object, which is a data structure holding a description of an operation to be performed, or of something that has happened and is being announced.

J

Java Java is a programming language originally developed by Sun, and used to develop Android applications.

It is important to note that while the Java language is used for Android applications, the Java bytecode and Java virtual machine are not. for more information, see the entry for Dalvik.

JDK Java Development Kit

Jellybean The code name for Android versions 4.1, 4.2 and 4.3

JNI Java Native Interface (wikipedia entry) is a programming framework that allows Java code to call or be called by "native" code (that is, code compiled in another language such as C, C++ or assembly).

K

KitKat The release name for Android version 4.4

L

Linux An open source operating system kernel, developed originally by Linus Torvalds, but over time by many thousands of developers worldwide.

Live-android A project to create an Android live-CD, for running Android on generic x86 platforms.

logcat A command to view messages in one of the system logs. See Android logger

M

manifest file See AndroidManifest.xml

mahimahi The machine name used in the kernel for the development board used for the Nexus One product.

MSM Mobile Station Modem. Chipset manufactured by Qualcomm. Can for instance be found in cell phones containing the Snapdragon chipsets (HTC Desire/Nexus One).

N

NDK Native Development Kit. A set of tools, build files and instructions to generate native code (usually libraries) to be used with Android systems. Native libraries are most often used as part of JNI (to allow Java code to call C code, or vice versa).

O

OpenGL ES 3D graphics system and API for Android applications

R

repo Android repository manager. This is a wrapper program (written in Python) over the git tool, for managing the multiple git repositories that make up the entire Android code base. See http://source.android.com/download/using-repo

rild Radio-Interface-Link daemon. This is the daemon which handles communication between the rest of the Android system and the "radio interface" (otherwise known as the phone portion of an Android-based mobile phone system). In the simulator, since the phone hardware is not present, there is a program which runs to simulate the radio interface.

S

Saphire

SL4A Scripting Layer for Android - an execution environment that let's users use scripting languages (such as Python or Ruby), instead of Java, to write programs for Android. See Android Scripting

SGL 2D graphics layer for Android applications

SQLite A powerful and lightweight relational database engine used by the Android system components, and available to all Android applications.

T

TARGET_PRODUCT An environment variable used by the build system to indicate the product that the software should be built for. This and other TARGET_* variables are set using the choosecombo() function in build/envsetup.sh. If not set, the TARGET_* variables will use defaults when you run the 'm' alias, after source-ing build/envsetup.sh into your shell environment. Otherwise, use the choosecombo() function to set them.

ex: $ cd mydroid ; source build/envsetup.sh ; choosecombo

The options for TARGET_PRODUCT depend on entries in the AndroidProducts.mk files under build/target/products and vendor/*/*/AndroidProducts.mk in your repository.

toolbox The name of a multi-function program in the Android system. This program contains code for the single program toolbox to act like several different programs and utilities. Normally 'toolbox' is stored in /system/bin, and is symlinked to other names. It uses argv[0] to determine which program to behave like, when run. It is very similar in this regard to 'busybox', which another multi-function program used in many other embedded Linux systems.

Trout ARM linux kernel machine ID for the HTC Dream hardware (used in the t-Mobile G1 and the ADP1)

See http://www.arm.linux.org.uk/developer/machines/list.php?id=1440

V

vold volume daemon - a process on an android system responsible for managing mounting and unmounting file system (volumes)

W

wakelocks A kernel mechanism for Android power management. When a thread holds a wakelock, the kernel will refrain from entering a low-power state.

Back to Android Portal

Z

zygote The first Dalvik virtual machine instance. All other java applications that are started in the system are spawned from zygote.

Category:

- Android

> From: eLinux.org

Android Tutorials

Here are some "getting started" tutorials for Android and Android development:

- http://vis.berkeley.edu/courses/cs160-sp08/wiki/index.php/Getting_Started_with_Android
- Simple HOWTO building some linux apps and libraries from scratch on android
    - Contains instructions for building add-ons like busybox, jupp (text editor), libFLAC, xvid, nmap and dropbear
- From Unboxing Panda to Building and Loading an App
    - Contains instructions for unboxing Panda, the hardware needed, where to get a pre-built Panda build, how to program it, how to install the Android tools and how to build and install the app on Panda. Also includes a link to the demo app, "DisableSuspend." This tutorial features builds from http://www.linaro.org, an industry consortium for improving ARM upstream support. It contains every step needed in one place.

Contents

YouTube AndroidDevelopers channel

There are numerous videos (including tutorials) at the Android Developers channel on YouTube.

See http://www.youtube.com/profile?user=androiddevelopers#g/u

Karim Yaghmour presentations

- Android Internals - Karim's presentations at Android Builders Summit 2011
- Porting Android to New Hardware - Karim's presentations at Android Builders Summit 2011
- Android For Embedded Linux Developers - Karim's presentation at AnDevCon 2011
- Understanding the Android System Server - Karim's presentation at AnDevCon 2011

Karim's courseware

Click on the "Courseware" thumbnail on the class page:

- http://www.opersys.com/training/embedded-android
- http://www.opersys.com/training/android-development
- http://www.opersys.com/training/embedded-linux
- http://www.opersys.com/training/linux-device-drivers

Michael Haim presentations

Also, Michael Haim has produced a large number of useful presentations about Android topics. These are available at: http://www.abelski.com/

You need to create an account to use these resources, but they are free for personal and academic use.

There are presenations available in the following categories:

- Android Fundamentals
- Android Workshops
- App Widgets Development
- Effective Programming
- Android Testing
- The Android Internals

Free Electrons Android Courseware

Free Electrons has released their complete Android training materials, under the usual Creative Commons BY-SA 3.0 license: http://free-electrons.com/blog/free-android-training-materials/ . It contains more than 400 pages of slides and practical labs.

There's a public git tree and a LaTeX source format making it easy to adapt the materials to your needs (if you are a trainer), to translate them and to contribute to them.

Category:

- Android

> From: eLinux.org

Android History

This page has a few tidbits regarding the history of Android

Contents

Overview

- Google's Open Source Android OS Will Free the Wireless Web Wired article, June 2008
    - Has a description of Rubin and Page's first meeting, and the overall Google strategy for Android
    - Some additional commentary (see the last 4 paragraphs) are at: The forgotten history of Android

2005

August, 2007 - Google acquires Android Inc.

- See Google Buys Android for Its Mobile Arsenal
    - Business Week, August 2005

2007

- See I, Robot: The Man Behind the Google Phone
    - New York Times article about Andy Rubin, Nov 2007

November, 2007 - Creation of Open Handset Alliance

2008

October, 2008 - The first phone, the G1, ships in the U.S.

2009

February, 2009 - The Open Embedded Software Foundation (OESF) is formed in Japan.

April, 2009 - 1.5 update (codenamed "CupCake")

July, 2009 - Mentor Graphics acquires Embedded Alley

September, 2009 - 1.6 SDK released (codenamed "Donut")

September-November, 2009 - Motorola, Samsung, HTC, Sony/Ericsson announce new Android phones

October, 2001 - 2.0 SDK released (codenamed "Eclair")

November, 2009 - Google announces Maps Navigation program

- Business analysis: Google Redefines Disruption: the "Less than Free" Business Model

2010

Category:

- Android

> From: eLinux.org

Android Versions

Contents

Versions

For a lot more detail, you should see the wikipedia page for Android version history

The following different Android versions have been released:

Version number	code name	API Level	Main features
1.1	*unknown*	2	The base release, with Android system, phone, camera, webkit-based browser, Google search, contacts, calendar, cloud synchronization, android market, etc.
1.5	Cupcake	3	camcorder mode, video and picture upload to net, auto bluetooth connect, animated screen transitions
1.6	Donut	4	voice search
2.1	Eclair	7	more screen resolutions, live wallpaper, MS exchange support, UI revamp
2.2	Froyo	8	Dalvik JIT, USB tethering, voice dialing, V8 and javascript, adobe flash support
4.4	KitKat	19	Low RAM improvements, sensor batching, full-screen UI, ART experimental test

See the wikipedia article#Update_history) for some good information about different versions.

Version 1.1

SDK released February, 2009

Version 1.5 (cupcake)

SDK released April, 2009

Kernel version: 2.6.27

Version 1.6 (donut)

SDK released September, 2009

Kernel version: 2.6.29

Version 2.1 (eclair)

SDK released October, 2009

Kernel version: 2.6.29

Version 2.2 (froyo)

SDK release May, 2010

Kernel version: 2.6.32

Version 4.4 (kitkat)

SDK release October 31, 2013

Kernel version: 3.10?

Links to Android 4.4 information

- Official Android Blog post:
  - http://officialandroid.blogspot.com/2013/10/android-for-all-and-new-nexus-5.html
- Consumer facing highlights:
  - http://www.android.com/versions/kit-kat-4-4/
- Platform highlights:
  - http://developer.android.com/about/versions/kitkat.html
- Android Developers Blog post:
  - http://android-developers.blogspot.com/2013/10/android-44-kitkat-and-updated-developer.html
- API highlights:
  - http://developer.android.com/about/versions/android-4.4.html
- SDK updates:
  - http://developer.android.com/tools/revisions/platforms.html

Fragmentation

Dealing with API levels

Developers should specify which API level their application requires or is known to work with.

See http://developer.android.com/guide/appendix/api-levels.html

Need to support multiple versions

Dianne Hackborn had this to say about fragmentation, after a developer complained about having to support different versions of Android:

Developers will never be able to rely on there only being one version to target. Never. Please drop that from your mind. This is not the case on Windows, it is not the case on MacOS X, it is not the case on any platform that is not uber-tightly controlled.

That would mean that every user, of every Android device, would be delivered free updates as long as they are using the device. This would completely prevent very interesting Android devices like the Dell Streak (which due to its interesting design requires some customizations to the platform), and would significantly limit the ability of manufacturers to push Android into other interesting places.

This would also very greatly slow down the development of the platform. Every release of Android would need to be extensively QAed on every existing Android device before it could go out to any of them. There are currently > 50 such devices, and that number is growing exponentially. Android just can't scale if that final productization and QA phase is not spread out to the manufacturers. A core part of what makes our Android model work is that most products are owned by their manufacturer.

Here's a very small example: you say that manufacturers would be responsible for the kernel. Obviously, that is required because the kernel needs to talk with whatever their hardware is. However, pretty much every release of the platform has snapped up to a more recent version of the Linux kernel. The QA for that version of the platform is done against that kernel on a small set of specific devices. For what you propose, it would either need to be QAed across all possible kernels, or require that all manufacturers upgrade to the newer kernel before it can go out to any devices.

I honestly think that developers who are demanding there be only one version they need to think about are living in a strange fantasy world. Windows developers have always needed to think about 3 or so active versions of windows. From the stats I have seen there are regularly two very active versions of MacOS X that users run. Heck, if you are a web developer you need to consider different browsers and different versions implementing different versions of web standards.

I can more understand the issues from users, who want to be able to upgrade their current device to a newer version. Even there, though, if you compare us against another company that say releases a major platform update every year... we're not that bad. Look at the devices that have been out... how many at this point haven't received a significant platform update in the year since they came out? I think not many -- very few of them have been around for more than a year, and at this point the bulk of those have received at least one update. The main difference is that Android has gone through 3-4 significant updates in a year... so sure, not every one of those goes to every device, but I do think the manufacturers are showing that they are going to do some work to snap their devices to a more recent version, and I also think that as they are learning about Android and going through this upgrade experience the first few times they are learning how to better handle it.

At the end of the day Android is not a single monolithic uniform environment. That isn't what Android is intended to be, and honestly I think that this is something that should be pretty clear from day one. I consider that one of Android's strengths.

Category:

- Android

Android Linux Kernel

Android use Linux as the low-level kernel and add lots of specific features for smart devices.

> From: eLinux.org

Android Kernel Download

See http://source.android.com/source/building-kernels.html

Most of following information is outdated.

Main Google Android Kernels

The main Google repository with Android source code is at: https://android.googlesource.com/

There are (as of July 2014) 4 main separate kernel repositories at that site:

- common
- exynos
- goldfish
- lk
- samsung
- tegra
- msm
- omap

To download one of these and use it directly, you can use git. For example: (Do not use the git protocol to clone)

```
git clone https://android.googlesource.com/kernel/common.git kernel
```

To preserve your sanity, it's probably worth downloading this into a 'kernel' directory in your overall Android source directory scheme

You can use repo, following the instructions at http://source.android.com/download, to pull down the entire Android source. However, when you download the rest of the Android source code, using the 'repo' command, you do NOT automatically get a kernel tree included. That is, a kernel git tree is not referenced in the default Android manifest file,

To add projects, such as the kernel, to your overall Android repository scheme, you add the appropriate kernel repository to your local manifest.xml file. This file is located in the .repo directory.

To include the kernel/common tree, include a line like this in .repo/manifest.xml:

```
<project path="kernel/common" name="kernel/common" />
```

The complete list of projects (including other kernel options besides kernel/common) is listed on https://android.googlesource.com/.

Note that the default revision for git repositories is specified in the \ tag in manifest.xml as "revision=master" but the kernel/common repository may not have a head called "master". In that case if you just type "repo sync kernel/common" you may see the message:

```
error: revision master in kernel/common not found
```

Typically the heads in the kernel/common repository will be called android-2.6.x (where x is the kernel number); specifying this number in the manifest should allow repo to sync properly, i.e.:

```
<project path="kernel/common" name="kernel/common" revision="android-2.6.27"/>
```

You can view the heads by clicking on the project link from http://android.git.kernel.org/.

For more about repo, see http://source.android.com/download/using-repo

Other Repositories with Android-specific changes

- Linux kernel for omap and beagle-board, by Embinux: http://labs.embinux.org/git/cgit.cgi/repo/kernel.git
  - clone with: git clone git://labs.embinux.org/repo/kernel.git kernel

'Raw' Android kernel patches

I do not know of any freely available patches for the Linux kernel with the Android fixes, as of November 2009. I have, however, heard of multiple efforts to extract the patches to make it easier to port the Android kernel features onto newer Linux kernels.

Here is a way of extracting raw Android patches at a certain point in time, though this may be dated:

```
(Do not use the git protocol to clone the source)
git clone https://android.googlesource.com/kernel/common.git android-kernel
cd android-kernel
git checkout --track -b android-2.6.32 origin/android-2.6.32
git fetch --tags git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-2.6.32.y.git
git shortlog v2.6.32.9..HEAD
git format-patch v2.6.32.9..HEAD
```

Sum total 173 patches for the 2.6.32.9 kernel as of writing.

If anyone knows where raw android kernel patches are available, please add a link here. See also the Android Kernel Features page for more information about individual kernel features.

Category:

- Android

> From: eLinux.org

Android Kernel Build

How to build an Android kernel.

The Android build system may or may not automatically rebuild a kernel for you.

This page documents how to build an Android kernel, independent of the regular Android "distribution" build system.

[FIXTHIS - nothing here yet.]

Building an Individual application

Steps required to compile individual application package in Android SDK.

1. Go to build directory under Android sdk directory
2. Execute source source envsetup.sh
3. Go to corresponding application directory
4. Issue mm command to build the application only

Category:

- Android

> From: eLinux.org

Android Kernel Installation

How to use a new kernel with the emulator or on a product.

In general, this will be very product-specific.

Contents

With the G1/ADP1

On the ADP1, you can boot using a specific kernel using fastboot, which will send it to the board as part of the bootup sequence. Specify the kernel as one of the parameters to the 'fastboot boot' option. See http://www.gotontheinter.net/content/fastboot-cheat-sheet for details.

Installing a boot image

You can create a "boot" image directly, and install that to the boot or recovery flash partition manually. This can be done either using a running kernel or using fastboot.

See these instructions for doing that: http://android-dls.com/wiki/index.php?title=HOWTO:_Unpack,_Edit,_and_Re-Pack_Boot_Images

Also, you can install the kernel into a recovery image, and install the recovery image using the system recovery procedure.

With the emulator

You can specify the kernel to use with the emulator using the '-kernel' command line option.

There are some interesting notes about using the emulator without specifying a AVD file at: http://groups.google.com/group/android-platform/browse_thread/thread/591290fde1e9865a

On a board using an SDCARD

OMAP EVM

For an OMAP EVM, using the Android image available from Mistral, place a new uImage on the SDCARD at the root the vfat partition. The bootloader will run the commands (from where??) to load the uImage and boot it.

Note that for the OMAP EVM, you need to have the board configured to boot from sdcard, rather than from flash.

Category:

- Android

> From: eLinux.org

Android Kernel Versions

Contents

Kernel versions known to run Android

2.6.23

The first-generation Google-TV products (released about November, 2010) used kernel version 2.6.23. It is speculated that this is due to existing binary driver support for the Intel chipsets used in those products.

2.6.25

The original (version 1.0) of Android for the G1/ADP1 used Linux version 2.6.25

2.6.27

The 1.5 release of Android (cupcake) for the G1/ADP1 used Linux version 2.6.27

2.6.29

As of September, 2009, the kernel/common.git tree for Android has a 2.6.29 kernel. Donut uses this kernel.

2.6.32

As of July 2010, the kernel/common.git tree for Android has a 2.6.32 kernel. This kernel is used by Froyo.

2.6.35

Gingerbread used kernel version 2.6.35.

Selecting a kernel tree for a new port

This article has good information on porting Android to a new device. Note that it debates which kernel tree is appropriate to start with, for porting to a new device:

- http://www.linuxfordevices.com/c/a/Linux-For-Devices-Articles/Porting-Android-to-a-new-device/

Android git kernel trees

As of July 2010, the android repository on kernel.org had the following kernel trees:

kernel/common.git This is the kernel tree that matches what is put into official Android products by Google or its partners

kernel/experimental.git some experimental stuff

kernel/linux-2.6.git mirror of Linus' kernel tree. This is a reference for the kernel tree used in Android

kernel/lk.git not a kernel tree, but a bootloader

kernel/msm.git kernel for MSM (Qualcomm chip used in many HTC products)

kernel/omap.git mirror of kernel tree maintained by texas instruments., supporting OMAP chips. usually based on kernel.org latest version, android products w OMAP chip can use this kernel port.

kernel/tegra.git nvidia kernel tree

Category:

- Android

> From: eLinux.org

Android Kernel Features

Contents

Kernel features unique to Android

In the course of development, Google developers made some changes to the Linux kernel. The amount of changes is not extremely large, and is on the order of changes that are customarily made to the Linux kernel by embedded developers (approximately 250 patches, with about 3 meg. of differences in 25,000 lines). The changes include a variety of large and small additions, ranging from the wholesale addition of a flash filesystem (YAFFS2), to very small patches to augment Linux security (paranoid networking patches).

Various efforts have been made over the past few years to submit these to changes to mainline (mostly by Google engineers, but also by others), with not much success so far.

Resources

A very good overview of the changes is available in a talk by John Stultz at ELC 2011. (The talk has a somewhat misleading name.)

- Android OS for Servers- John Stultz, ELC 2011
  - This talks breaks down the differences between an Android Linux kernel and a stock Linux kernel, and provides information about the features of each.
- http://www.lindusembedded.com/blog/2010/12/07/android-linux-kernel-additions/
  - Lindus Embedded (Alex Gonzalez) has a listing of kernel changes based on an Android kernel for the Freescale MX51 SOC, with some good information about each change.
- http://yidonghan.wordpress.com/2010/01/28/porting-android-to-a-new-device/
  - Peter McDermott's excellent description of his work to port Android to the Nokia N810.
  - Also, see his annotated list of modified and added kernel files, at: http://www.linuxfordevices.com/files/misc/porting-android-to-a-new-device-p3.html
  - See the project site at sourceforge.

- http://www.slideshare.net/jollen/android-os-porting-introduction
  - Jollen Chen's excellent presentation on system-level Android features, including an overview of kernel features unique to Android: *Note: Parts of the presentation are in Chinese*

Temporary including in mainline 'staging'

Some changes were temporarily added at the "staging" driver area in the stock kernel, but were removed due to lack of support. See Greg KH blog post on -staging for 2.6.33, where he announces the **removal** of various Android drivers from -staging.

Android mainlining project

Several groups and individuals are working to get kernel changes from Android mainlined into the Linux kernel.

Please see Android Mainlining Project for more information.

List of kernel features unique to Android

Here is a list of changes/addons that the Android Project made to the linux kernel. As of September, 2011, these kernel changes are not part of the standard kernel and are only available in the Android kernel trees in the Android Open Source project.

This list does not include board- or platform-specific support or drivers (commonly called "board support").

Binder

Binder is an Android-specific interprocess communication mechanism, and remote method invocation system.

See Android Binder

ashmem

- ashmem - Android shared memory
  - implementation is in `mm/ashmem.c`

According to the Kconfig help "The ashmem subsystem is a new shared memory allocator, similar to POSIX SHM but with different behavior and sporting a simpler file-based API."

Apparently it better-supports low memory devices, because it can discard shared memory units under memory pressure.

To use this, programs open /dev/ashmem, use mmap() on it, and can perform one or more of the following ioctls:

- ASHMEM_SET_NAME
- ASHMEM_GET_NAME
- ASHMEM_SET_SIZE
- ASHMEM_GET_SIZE
- ASHMEM_SET_PROT_MASK
- ASHMEM_GET_PROT_MASK
- ASHMEM_PIN
- ASHMEM_UNPIN
- ASHMEM_GET_PIN_STATUS
- ASHMEM_PURGE_ALL_CACHES

From a thread on android-platform source

You can create a shared memory segment using:

```
    fd = ashmem_create_region("my_shm_region", size);
    if(fd < 0)
        return -1;
    data = mmap(NULL, size, PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
    if(data == MAP_FAILED)
        goto out;
```

In the second process, instead of opening the region using the same name, for security reasons the file descriptor is passed to the other process via binder IPC.

The libcutils interface for ashmem consists of the following calls: (found in system/core/include/cutils/ashmem.h)

- int ashmem_create_region(const char *name, size_t size);
- int ashmem_set_prot_region(int fd, int prot);
- int ashmem_pin_region(int fd, size_t offset, size_t len);
- int ashmem_unpin_region(int fd, size_t offset, size_t len);
- int ashmem_get_size_region(int fd);

pmem

- PMEM - Process memory allocator
  - implementation at: `drivers/misc/pmem.c` with include file at: `include/linux/android_pmem.h`
  - Brian Swetland says:

```
The pmem driver is used to manage large (1-16+MB) physically contiguous
regions of memory shared between userspace and kernel drivers (dsp, gpu,
etc).  It was written specifically to deal with hardware limitations of
the MSM7201A, but could be used for other chipsets as well.  For now,
you're safe to turn it off on x86.
```

David Sparks wrote the following: source

```
2. ashmem and pmem are very similar. Both are used for sharing memory
between processes. ashmem uses virtual memory, whereas pmem uses
physically contiguous memory. One big difference is that with ashmem,
you have a ref-counted object that can be shared equally between
processes. For example, if two processes are sharing an ashmem memory
buffer, the buffer reference goes away when both process have removed
all their references by closing all their file descriptors. pmem
doesn't work that way because it needs to maintain a physical to
virtual mapping. This requires the process that allocates a pmem heap
to hold the file descriptor until all the other references are closed.

3. You have the right idea for using shared memory. The choice between
ashmem and pmem depends on whether you need physically contiguous
buffers. In the case of the G1, we use the hardware 2D engine to do
scaling, rotation, and color conversion, so we use pmem heaps. The
emulator doesn't have a pmem driver and doesn't really need one, so we
use ashmem in the emulator. If you use ashmem on the G1, you lose the
hardware 2D engine capability, so SurfaceFlinger falls back to its
software renderer which does not do color conversion, which is why you
see the monochrome image.
```

logger

- logger - system logging facility
  - This is the kernel support for the 'logcat' command
  - The kernel driver for the serial devices for logging are in the source code `drivers/misc/logger.c`
  - See Android logger for more information about the kernel code
  - See Android Logging System for an overview of the system it supports

wakelocks

- wakelock - used for power management files `kernel/power/wakelock.c`
    - Holds machine awake on a per-event basis until wakelock is released
    - See Android Power Management for detailed information

oom handling

- oom handling modifications
    - lowmem notifications
    - implementation at: `drivers/misc/lowmemorykiller.c`
    - also at: `security/lowmem.c`

Informally known as the Viking Killer, the OOM handler simply kills processes as available memory becomes low. The kernel module follows rules for this that are supplied from user space in two ways:

1. init writes information about memory levels and associated classes:

2. The write value must be consistent with the above properties.

3. Note that the driver only supports 6 slots, so we have combined some of the classes into the same memory level; the associated processes of higher classes will still be killed first.
    - From /init.rc:

```
write /sys/module/lowmemorykiller/parameters/adj 0,1,2,4,7,15
write /sys/module/lowmemorykiller/parameters/minfree 2048,3072,4096,6144,7168,8192
```

1. User space sets the oom_adj of processes to put them in the correct class for their current operation. This redefines the meaning of oom_adj from that used by the standard OOM killer to something that is more aggressive and controlled.

These oom_adj levels end up being based on the process lifecycle defined here:
http://developer.android.com/guide/topics/fundamentals.html#proclife

Alarm timers

This is the kernel implementation to support Android's AlarmManager. It lets user space tell the kernel when it would like to wake up, allowing the kernel to schedule that appropriately and come back (holding a wake lock) when the time has expired regardless of the sleep state of the CPU.

POSIX Alarm Timers

Note that POSIX Alarm timers, which implement this functionality (but not identically), was accepted into mainline Linux in kernel version 3.0.

See Waking Systems from Suspend and http://lwn.net/Articles/439364/

paranoid network security

- paranoid network security
    - See Android_Security#Paranoid_network-ing

timed output / timed gpio

Generic gpio is a mechanism to allow programs to access and manipulate gpio registers from user space.

Timed output/gpio is a system to allow chaning a gpio pin and restore it automatically after a specified timeout. See `drives/misc/timed_output.c` and `drives/misc/timed_gpio.c` This expose a user space interface used by the vibrator code.

On ADP1, there is a driver at:

```
# cd /sys/bus/platform/drivers/timed-gpio
# ls -l
--w-------    1 0        0            4096 Nov 13 02:11 bind
lrwxrwxrwx    1 0        0               0 Nov 13 02:11 timed-gpio -> ../../../../devices/platform/timed-gpio
--w-------    1 0        0            4096 Nov 13 02:11 uevent
--w-------    1 0        0            4096 Nov 13 02:11 unbind
```

Also, there is a device at:

```
# cd /sys/devices/platform/timed-gpio
# ls -lR
.:
lrwxrwxrwx    1 0        0               0 Nov 13 01:34 driver -> ../../../bus/platform/drivers/timed-gpio
-r--r--r--    1 0        0            4096 Nov 13 01:34 modalias
drwxr-xr-x    2 0        0               0 Nov 13 01:34 power
lrwxrwxrwx    1 0        0               0 Nov 13 01:34 subsystem -> ../../../bus/platform
-rw-r--r--    1 0        0            4096 Nov 13 01:34 uevent

./power:
-rw-r--r--    1 0        0            4096 Nov 13 01:34 wakeup
```

RAM_CONSOLE

This allows saving the kernel printk messages to a buffer in RAM, so that after a kernel panic they can be viewed in the next kernel invocation, by accessing /proc/last_kmsg.

[Would be good to get more details on how to set this up and use it here!] [I guess this is something like pramfs?]

other kernel changes

Here is a miscellaneous list of other kernel changes in the mistral Android kernel:

- switch events - drivers/switch/* userspace support for monitoring GPIO via sysfs/uevent used by vold to detect USB
- USB gadget driver for ADB - drivers/usb/gadget/android.c
- yaffs2 flash filesystem
- support in FAT filesystem for FVAT_IOCTL_GET_VOLUME_ID
- and more...

Kernel configuration options

The file Documentation/android.txt has a list of required configuration options for a kernel to support an Android system.

Category:

- Android

> From: eLinux.org

Android Board Support

Porting Android to a new platform can be a challenge. Here are some resources to start with that:

Contents

Processor Support

ARM

Most ports of Android are to ARM-based platforms.

OMAP

- Mentor Graphics and Texas Instruments support Android on OMAP processors via the project
- See also Android on OMAP, which has a very thorough listing of issues faced in initially porting Android to OMAP

Mips

Mentor Graphics did a port of Android to MIPS.

See http://www.mipsandroid.org

(Unfortunately, this site requires registration.)

- MIPS now has support for SMP on Android - see MIPS Supports Symmetric Multiprocessing on Android Platform - Posted by Ken Cheung in IP Cores on Tuesday, June 1, 2010

x86

There is a whole well-developed project for Android on x86.

See http://www.android-x86.org/

At least one major product (Sony Internet TV) is reported to be x86-based. Intel has a team of developers working on Android issues. See Mark Gross' presentation from ELC 2010 for some tips from them about using Android

- Experiences in Android Porting, Lessons Learned,Tips and Tricks by Mark Gross, April 2010, Embedded Linux Conference 2010

- Intel is working on "native" Android support. See Intel prepping x86 port for Android 2.2 By Eric Brown, lwn.net, 2010-06-28

Individual Platform Support

For Nexus One

Unlocking the phone

Bryan Swetland says: (here)

All Nexus One devices have an unlockable bootloader (% fastboot oem unlock), which, once unlocked will allow you to reflash the boot partition (kernel + ramdisk), system partition, etc.

Full kernel sources are available here: http://android.git.kernel.org/?p=kernel/msm.git;a=shortlog;h=refs/heads/android-msm-2.6.29-nexusone (make mahimahi_defconfig to configure just like the production kernel). We're in the process of rebasing up to .32, on our way to .33 and beyond.

Serial port access

There are some serial port pins available on the micro-USB connector, which you can access if you have the right hardware.

See this discussion on xda-developers for detailed information and links.

Brian Swetland says: TTL level (~3.3v?) serial is present on the D+/D- pins of the micro USB connector whenever VBUS (usb +5v power) is not present. This is physical UART1 (ttyMSM0). In standard builds the FIQ kernel debugger runs there. You'll have to disable the FIQ debugger and enable the serial device in your kernel config if you want to use it as a regular serial port.

System Requirements

RAM (>256M if possible)

Dianne Hackborn had this to say (in August, 2009) about RAM requirements for Android:

I would recommend at least 128MB available to the *kernel*. In many architectures, a big chunk of RAM will be dedicated to the radio, so you need to take that into account, and the 128MB recommendation does not cover that. Also if your architecture allocates graphics surfaces in user space, bump it up by 16MB or so (The Qualcomm devices I have experience do their allocations in RAM outside of that accessible to the kernel.) And of course this also depends on the size and density of your screen, camera megapixels, etc. If the screen has more pixels than 320x480 or the camera is more than 3 megapixels, bump the size up accordingly.

For reference, the myTouch has 192MB of RAM which on 1.6 only left 100MB left for the kernel and user space, and was *very* tight on RAM. Don't go that low. An update to the Qualcomm radio apparently frees up a bunch of space, adding over 10MB or so to user space... that should run even 2.2 okay. At this level, though, the amount of RAM is probably the most important aspect to how well the device will run. Don't skip on RAM, and you'll have a much better running device, with a lot fewer headaches as you try to get everything working well. That is from painful experience. :)

Another reference point -- the Droid has 256MB RAM, which runs the system well, but it also does its graphics allocations in user space and has a high density screen so you can still end up not keeping as many processes running as you'd like if loading large pages with the browser, running lots of background services, etc.

The Nexus One has 512MB of RAM and honestly that is really more than we know what to do with. It is great. :) I ended up putting some code into the activity manager to put a hard limit on the number of processes we would keep around, because there was so much memory we had often could keep way more processes than was useful. That was never an issue on

Droid. ;)

Category:

- Android

Android System Information

About Android System, from system booting, power management, security, Dalvik VM, packages, networking, file systems to Android logging system.

> From: eLinux.org

Android Booting

The bootup of an Android system consists of several phases, which are outlined here.

Contents

Key bootup components

Bootloader

The first program which runs on any Android system is the bootloader. Technically, the bootloader is outside the realm of Android itself, and is used to do very low-level system initialization, before loading the Linux kernel. The kernel then does the bulk of hardware, driver and file system initialization, before starting up the user-space programs and applications that make up Android.

Often, the first-stage bootloader will provide support for loading recovery images to the system flash, or performing other recovery, update, or debugging tasks.

The bootloader on the ADP1 detects certain keypresses, which can be used to make it load a 'recovery' image (second instance of the kernel and system), or put the phone into a mode where the developer can perform development tasks ('fastboot' mode), such as re-writing flash images, directly downloading and executing an alternate kernel image, etc.

'init'

A key component of the Android bootup sequence is the program 'init', which is a specialized program for initializing elements of the Android system. Unlike other Linux systems (embedded or otherwise), Android uses its own initialization program. (Linux desktop systems have historically used some combination of /etc/inittab and sysV init levels - e.g. /etc/rc.d/init.d with symlinks in /etc/rc.d/rc.[2345]). Some embedded Linux systems use simplified forms of these -- such as the init program included in busybox, which processes a limited form of /etc/inittab, or a direct invocation of a shell script or small program to do fixed initialization steps.

The Android 'init' program processes two files, executing the commands it finds in them, called 'init.rc' and 'init.\.rc', where \ is the name of the hardware that Android is running on. (Usually, this is a code word. The name of the HTC1 hardware for the ADP1 is 'trout', and the name of the emulator is 'goldfish'.

The 'init.rc' file is intended to provide the generic initialization instructions, while the 'init.\.rc' file is intended to provide the machine-specific initialization instructions.

'init' resources

The syntax for these .rc files is documented in a readme file in the source tree. See the Android init language reference

Or, see also: kandroid copy of old Android PDK

- Note that the old PDK has been retracted from where it used to found below http://source.android.com/porting
  - You may be able to reconstruct it though:
    - You need a local copy of the AOSP sourcetree, and run the usual *build/envsetup.sh* preparation
    - Use *repo init -b* to check out the AOSP sourcetree with a tag around 2.3, then *make sdk sdk_all*
    - This worked for me, though I used some tag around 4.0.4. Some links had to be fixed in the resulting html output

See also http://www.androidenea.com/2009/08/init-process-and-initrc.html

Sequence of boot steps on ADP1

firmware

- first-stage bootloader runs
  - it detects if a special key is held, and can launch the recovery image, or the 'fastboot' bootloader
- eventually, a kernel is loaded into RAM (usually with an initrd)
  - normally, this will be the kernel from the 'boot' flash partition.

kernel

- the kernel boots
  - core kernel initialization
    - memory and I/O areas are initialized
    - interrupts are started, and the process table is initialized
  - driver initialization
  - kernel daemons (threads) are started
  - root file system is mounted
  - the first user-space process is started
    - usually /init (note that other Linux systems usually start /sbin/init)

user space

- the kernel runs /init
  - /init processes /init.rc and /init.\.rc
  - dalvik VM is started (zygote). See Android Zygote Startup
  - several daemons are started:
    - rild - radio interface link daemon
    - vold - volume daemon (media volumes, as in file systems - nothing to do with audio volume)
- the system_server starts, and initializes several core services
  - See http://www.androidenea.com/2009/07/system-server-in-android.html
  - initalization is done in 2 steps:
    - 1) a library is loaded to initialize interfaces to native services, then

        - 2) java-based core services are initialized in ServerThread::run() in SystemServer.java
- the activity manager starts core applications (which are themselves dalvik applications)
    - com.android.phone - phone application
    - android.process.acore - home (desktop) and a few core apps.
- other processes are also started by /init, somewhere in there:
    - adb
    - mediaserver
    - dbus-daemon
    - akmd

Tools for analyzing Android Bootup

- logcat - see Android Logging System
    - try this command: 'adb logcat -d -b events | grep "boot"

```
01-01 00:00:08.396 I/boot_progress_start(  754): 12559
01-01 00:00:13.716 I/boot_progress_preload_start(  754): 17879
01-01 00:00:24.380 I/boot_progress_preload_end(  754): 28546
01-01 00:00:25.068 I/boot_progress_system_run(  768): 29230
01-01 00:00:25.536 I/boot_progress_pms_start(  768): 29697
01-01 00:00:25.958 I/boot_progress_pms_system_scan_start(  768): 30117
01-01 00:00:40.005 I/boot_progress_pms_data_scan_start(  768): 44171
01-01 00:00:45.841 I/boot_progress_pms_scan_end(  768): 50006
01-01 00:00:46.341 I/boot_progress_pms_ready(  768): 50505
01-01 00:00:49.005 I/boot_progress_ams_ready(  768): 53166
01-01 00:00:52.630 I/boot_progress_enable_screen(  768): 56793
```

-
    - or this: 'adb logcat -d | grep preload'

```
10-15 00:00:17.748 I/Zygote  (  535): ...preloaded 1873 classes in 2438ms.
10-15 00:00:17.764 I/Zygote  (  535): ...preloaded 0 resources in 0ms.
10-15 00:00:17.772 I/Zygote  (  535): ...preloaded 15 resources in 7ms.
```

- Bootchart - see Using Bootchart on Android
- strace is pretty handy also, to see the timings for system calls from a process as it runs
    - You can use strace as a wrapper for a program in init.rc, and save the results to a file
    - Use -f to follow sub-processes
    - Use -tt to get detailed timestamps for syscalls
    - Use -o to output the data to a file

Here is an example of using strace to follow the startup of zygote, and the apps that are forked from it.

Replace:

```
service zygote /system/bin/app_process -Xzygote /system/bin --zygote --start-system-server
```

with

```
service zygote /system/xbin/strace -f -tt -o /cache/debug/boot.strace /system/bin/app_process -Xzygote /system/bin --
zygote --start-system-server
```

Here is some sample data:

```
$ head boot.strace
571   00:00:11.389939 execve("/system/bin/app_process", ["/system/bin/app_process", "-Xzygote", "/system/bin", "--zyg
ote", "--start-system-server"], [/* 15 vars */]) = 0
571   00:00:11.658878 brk(0)            = 0x804b000
571   00:00:11.659048 mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x77f9a000
571   00:00:11.659169 readlink("/proc/self/exe", "/system/bin/app_process", 4096) = 23
571   00:00:11.659339 access("/etc/ld.so.preload", R_OK) = 0
571   00:00:11.659440 open("/etc/ld.so.preload", O_RDONLY) = 3
571   00:00:11.659548 fstat64(0x3, 0x7fa76650) = 0
571   00:00:11.659887 mmap2(NULL, 36, PROT_READ|PROT_WRITE, MAP_PRIVATE, 3, 0) = 0x77f99000
571   00:00:11.659970 close(3)          = 0
571   00:00:11.660071 open("/lib/libc_sse.so", O_RDONLY) = 3
```

Please note that writing the strace data takes extra time. For long sequences of very fast syscalls (such as when the timezone file is being read) the overhead of strace itself exaggerates the timings in the trace. Use the timing information with caution.

Notes on the Android startup procedure

Overview

See "Android Initialization Process" at: http://blog.chinaunix.net/u2/85805/showart_1421736.html (this address is not work), using http://blog.chinaunix.net/space.php?uid=7788581&do=blog&id=2558375 instead.

strace

http://benno.id.au/blog/2007/11/18/android-runtime-strace

Interaction of different processes on application initialization

Talking about Android Process - http://blog.csdn.net/mawl2002/archive/2009/06/24/4295905.aspx

Improving Bootup Time presentation

See Improving Android Boot Time - notes and material for a talk at LinuxCon North America, 2010 by Tim Bird

News

1-second boot of Android

Ubiquitous Corporation has announced boot of ARM-based Android system in 1 second. Actually, it's more like a suspend and resume than a boot. See http://www.linuxfordevices.com/c/a/News/Ubiquitous-QuickBoot/?kc=LNXDEVNL032410 [March, 2010]

Category:

- Android

> From: eLinux.org

Android Power Management

Notes on Power Management in Android

Contents

wakelocks

The first version of Android utilized a system called "wakelocks", which was a set of patches to the Linux kernel to allow a caller to prevent the system from going to low power state.

Each wakelock is defined with a name and type. The type is one of:

- WAKE_LOCK_IDLE, or
- WAKE_LOCK_SUSPEND.

The name is an arbitrary ASCII string.

When a wake lock of type "IDLE" is in effect, the system will not enter idle (low power) state, and this should make the device more responsive. That is, it does not have to wake up from idle to respond to interrupts or events. When a wake lock of type "SUSPEND" is held, then the system will not suspend, which takes even longer to resume from.

You can, from user space, see the currently defined wakelocks and a bunch of information about their status, using 'cat /proc/wakelocks'

Below is an example of output from that command.

To see the code for this, see the file:

```
drivers/android/power.c
```

NOTE: See the Discussion page for a note on this file.

Creating a wakelock

Kernel code can define a wakelock, and get a handle to it, by calling:

```
#include <linux/wakelock.h>
wake_lock_init(struct wakelock *lock, int type, const char *name);
```

From user space, a process can write a name to

```
/sys/power/wake_lock
```

to create and lock a suspend lock with that name.

Using a wakelock inside the kernel

Kernel code can acquire and release the lock with one of the following:

```
void wake_lock(struct wake_lock *lock);
void wake_unlock(struct wake_lock *lock);
```

Kernel code can also set a timeout to specify automatic release of the wakelock after a specific period, with:

```
void wake_lock_timeout(struct wake_lock *lock, long timeout);
```

The wakelock does not need to be held prior to calling this (it will automatically lock the wakelock and register the timeout).

Using a wakelock from user space

To release a suspend wake lock from user space, a process can write the lock name to: /sys/power/wake_unlock

Sample 'cat /proc/wakelocks' output

Here are some wakelocks from my ADP1 phone, running Donut (I think):

Note: I widened the columns and adjusted the title row to make the columns line up better.

```
 $ cat /proc/wakelocks
name                    count   expire_count  active_since
                                        wake_count          total_time      sleep_time      max_time        last_chan
ge
"PowerManagerService"   3580    0       0     0             1706674438454   1354421173104   62626251221     317019369
96687
"mddi_link_active_idle_lock" 4641 0     0     0             26749877925     0               253234863       317019037
32527
"msmfb_idle_lock"       2549    0       0     0             28889923076     0               266601563       317019026
33894
"SMD_RPCCALL"           8913    0       34    0             9038391159      6843658471      176025391       316903165
43807
"audio_pcm_idle"        12      0       0     0             68230224609     0               27459259033     316903155
67244
"audio_pcm"             12      0       0     0             68230285646     12423248292     27459289551     316903155
67244
"rpc_read"              60      0       0     0             3112792         823974          91553           316903150
48445
"adsp"                  12      0       0     0             67035552978     11988464355     27387664794     316902827
91365
"rpc_read"              12      0       0     0             671387          213624          91553           316865822
29842
"usb_mass_storage"      0       0       0     0             0               0               0               0
"rpc_read"              137     0       0     0             8972160         6744383         122070          316857884
67635
"rpc_server"            277     0       0     0             406372060       320922839       41412354        316857878
87801
"rpc_read"              277     0       0     0             117919940       111480725       94848633        316857878
26766
"alarm"                 1220    0       0     0             27851074219     25716247547     382141114       316692709
50545
"rpc_read"              4909    0       0     0             375915506       347930899       16021729        313764270
78475
"gpio_input"            0       0       0     0             0               0               0               0
"mddi_idle_lock"        54      0       0     0             8752532958      0               249603271       311541907
19832
```

```
"SMD_DATA5"           862     862     377   0             620518382572   567607416986   6926910400     311534217
37899
"alarm_rtc"           177     10      0     0             47826019299    47825439463    1801544190     311524716
95175
"KeyEvents"           13271   0       0     0             11482665934    343322752      503204346      305258862
45957
"evdev"               4437    0       0     0             4980529833     98724365       506927490      305258848
11631
"evdev"               188     0       6     0             3093719493     2680358890     393676758      305229187
77697
"rpc_read"            18      0       0     0             1464840        0              335693         305204255
52599
"gpio_kp"             68      0       14    0             14835723878    2092010498     933135987      304450590
05968
"evdev"               52      0       0     0             98999026       0              24139405       304381280
67247
"rpc_read"            596     0       0     0             745910654      211212166      491180420      300462090
60900
"rpc_read"            1331    0       0     0             286651593      169342025      112518310      300462035
98253
"rpc_read"            10      0       0     0             1495361        427247         274658         359698045
3801
"qmi0"                2       2       0     0             996235351      0              501717529      173250290
527
"qmi2"                1       1       0     0             490385742      0              490385742      173250290
527
"qmi1"                1       1       0     0             493193359      0              493193359      173250290
527
"evdev"               0       0       0     0             0              0              0              0
"evdev"               0       0       0     0             0              0              0              0
"rpc_read"            2       0       0     0             427247         0              274659         746135009
7
"mt9t013"             0       0       0     0             0              0              0              0
"gpio_input"          0       0       0     0             0              0              0              0
"gpio_input"          0       0       0     0             0              0              0              0
"SMD_DATA7"           0       0       0     0             0              0              0              0
"SMD_DATA6"           0       0       0     0             0              0              0              0
"msm_serial_hs_rx"    0       0       0     0             0              0              0              0
"unknown_wakeups"     0       0       0     0             0              0              0              0
"deleted_wake_locks"  36      0       0     0             2593991        518798         1068114        0
"radio-interface"     329     0       0     549499512     437556091307   299558441157   2895507812     317015406
95418
"vbus_present"        3       2       0     16399444580   26400295008545 25248585083008 25852771240234 316856907
80867
"main"                27      0       0     547991455078  4674244259033  0              627599700928   311540988
00887
"mmc_delayed_work"    796     796     172   0             405246968975   396150246568   689067383      311535569
91805
"SMD_DS"              682     681     192   558929443     486005943591   268692961407   2368892822     317015313
57039
```

Patch submission controversy

Arve Hjønnevå (of Google) sent patches to the linux-pm mailing list in of 2009, but they were rejected. See this thread for the submission and resulting discussion. This was the third version of the patches submitted for review by the kernel community.

The reasons for the rejection are described in the LWN.NET article Wakelocks and the embedded problem.

Wakelock documentation (from patch)

The 3rd version of the wakelock patch included the following /Documentation/power/wakelock.txt file

```
Wakelocks
=========
```

```
A locked wakelock, depending on its type, prevents the system from entering
suspend or other low-power states. When creating a wakelock, you can select
if it prevents suspend or low-power idle states.  If the type is set to
WAKE_LOCK_SUSPEND, the wakelock prevents a full system suspend. If the type
is WAKE_LOCK_IDLE, low-power states that cause large interrupt latencies, or
that disable a set of interrupts, will not be entered from idle until the
wakelocks are released. Unless the type is specified, this document refers
to wakelocks with the type set to WAKE_LOCK_SUSPEND.

If the suspend operation has already started when locking a wakelock, it will
abort the suspend operation as long it has not already reached the suspend_late
stage. This means that locking a wakelock from an interrupt handler or a
freezeable thread always works, but if you lock a wakelock from a suspend_late
handler you must also return an error from that handler to abort suspend.

Wakelocks can be used to allow user-space to decide which keys should wake the
full system up and turn the screen on. Use set_irq_wake or a platform specific
api to make sure the keypad interrupt wakes up the cpu. Once the keypad driver
has resumed, the sequence of events can look like this:
- The Keypad driver gets an interrupt. It then locks the keypad-scan wakelock
  and starts scanning the keypad matrix.
- The keypad-scan code detects a key change and reports it to the input-event
  driver.
- The input-event driver sees the key change, enqueues an event, and locks
  the input-event-queue wakelock.
- The keypad-scan code detects that no keys are held and unlocks the
  keypad-scan wakelock.
- The user-space input-event thread returns from select/poll, locks the
  process-input-events wakelock and then calls read in the input-event device.
- The input-event driver dequeues the key-event and, since the queue is now
  empty, it unlocks the input-event-queue wakelock.
- The user-space input-event thread returns from read. It determines that the
  key should not wake up the full system, releases the process-input-events
  wakelock and calls select or poll.

                 Key pressed   Key released
                     |             |
keypad-scan          ++++++++++++++++++
input-event-queue        +++          +++
process-input-events       +++          +++


Driver API
==========

A driver can use the wakelock api by adding a wakelock variable to its state
and calling wake_lock_init. For instance:
struct state {
    struct wakelock wakelock;
}

init() {
    wake_lock_init(&state->wakelock, WAKE_LOCK_SUSPEND, "wakelockname");
}

Before freeing the memory, wake_lock_destroy must be called:

uninit() {
    wake_lock_destroy(&state->wakelock);
}

When the driver determines that it needs to run (usually in an interrupt
handler) it calls wake_lock:
    wake_lock(&state->wakelock);

When it no longer needs to run it calls wake_unlock:
    wake_unlock(&state->wakelock);

It can also call wake_lock_timeout to release the wakelock after a delay:
    wake_lock_timeout(&state->wakelock, HZ);
```

```
This works whether the wakelock is already held or not. It is useful if the
driver woke up other parts of the system that do not use wakelocks but
till need to run. Avoid this when possible, since it will waste power
if the timeout is long or may fail to finish needed work if the timeout is
short.
```

Motorola quickwakeup feature

Jocelyn Falempe of Motorola proposed (in November, 2009) a quickwakeup feature to make it possible to reduce the time for a periodic job to resume, do a small amount of work, and suspend again.

http://patchwork.kernel.org/patch/58064/

From the patch:

```
The purpose of this feature is to drastically reduce the suspend/resume
time for device driver which needs to do periodic job.
In our use case (android smartphone), the system is most of the time in
suspend to RAM, and needs to send a low level command every 30s. With
current framework it takes about 500ms on omap3430 to resume the full
system, and then suspend again. With quickwakup feature, in the resume
process after resuming sysdev and re-enabling irq, the driver handler is
executed, and then it suspends again.
This new path takes 20ms for us, which leads to good power-saving.
```

Earlysuspend

Arve's patches also included something referred to as "earlysuspend", but I haven't reviewed this yet to see what it is.

May 2010 patch submission - "Suspend blockers"

Arve refactored the patches and did a name change, and submitted them again in April and May of 2010. There was a LOT of discussion on the linux-pm mailing list, which many developers participated in. The discussion raised lots of questions, and lots of responses were given. People interested in Android power management may get more enlightenment by reading the thread.

http://groups.google.com/group/linux.kernel/browse_frm/thread/b6fed7e38365c259/c92d8b4a41f87902?hl=en&tvc=1&q=linux.kernel+suspend+block+api+(version+6)#c92d8b4a41f87902#c92d8b4a41f87902)

Category:

- Android

> From: eLinux.org

Android Security

Contents

Overview

The overall architecture of Android security is described at: http://developer.android.com/guide/topics/security/security.html

Each application is given its own Linux user id (UID) and group ID

Kernel-level security

Users and groups

Each application is assigned a uid and gid at install time. Application data files are stored in /data/data/\..., and are read-writable only by that application process.

Sample File Permissions

Here is an example from my ADP1 phone (lots of lines omitted to reduce noise):

(Oh, and yes, I'm using busybox - find, xargs, and sort are not available otherwise)

```
# find /data/data -type f | xargs ls -l | sort -k3 -n
-rw-------    1 1000     1000         1954 Nov 12 01:10 /data/data/com.android.providers.subscribedfeeds/files/sslcac
he/android.clients.google.com.443
-rw-r--r--    1 1000     1000       147608 Apr  6  2009 /data/data/com.google.tts/lib/libspeechsynthesis.so
-rw-rw----    1 1000     1000           65 Nov  5 02:01 /data/data/com.google.android.systemupdater/shared_prefs/syst
em_update_helper.xml
-rw-rw----    1 1000     1000          679 Nov 11 23:18 /data/data/com.android.settings/shared_prefs/com.android.sett
ings_preferences.xml
-rw-rw----    1 1000     1000         2000 May 14 20:07 /data/data/com.google.android.location/files/DATA_Preferences
-rw-rw----    1 1000     1000         6144 Dec 19  2008 /data/data/com.android.settings/databases/webviewCache.db
-rw-rw----    1 1000     1000        11264 Nov 12 01:10 /data/data/com.android.providers.subscribedfeeds/databases/su
bscribedfeeds.db
```

```
-rw-rw----    1 1000     1000        14336 Dec 19  2008 /data/data/com.android.settings/databases/webview.db
-rw-rw----    1 1000     1000        36864 Nov 12 18:23 /data/data/com.android.providers.settings/databases/settings.
db
-rw-rw----    1 1000     1000       129024 Nov 12 18:45 /data/data/com.google.android.server.checkin/databases/checki
n.db
-rw-rw-r--    1 1000     1000          120 Nov 12 01:09 /data/data/com.android.providers.subscribedfeeds/shared_prefs
/subscribedFeeds.xml
-rwxrwx---    1 1000     1000        54052 Dec 20  2008 /data/data/com.android.settings/files/wallpaper
-rw-------    1 1001     1001            4 Oct 31 21:09 /data/data/com.android.providers.telephony/app_parts/PART_125
7023388570
-rw-------    1 1001     1001            4 Oct 31 21:10 /data/data/com.android.providers.telephony/app_parts/PART_125
7023445796

...
-rw-rw----    1 1001     1001          103 May 13  2009 /data/data/com.android.providers.telephony/shared_prefs/prefe
rred-apn.xml
-rw-rw----    1 1001     1001          122 Oct 28 17:37 /data/data/com.android.phone/shared_prefs/com.android.phone_p
references.xml
-rw-rw----    1 1001     1001          126 Sep  3  2008 /data/data/com.android.phone/shared_prefs/_has_set_default_va
lues.xml
-rw-rw----    1 1001     1001         7168 Nov  5 02:01 /data/data/com.android.providers.telephony/databases/telephon
y.db
-rw-rw----    1 1001     1001        69632 Nov  6 01:58 /data/data/com.android.providers.telephony/databases/mmssms.d
b
-rw-rw----    1 10000    10000         114 Apr 20  2009 /data/data/com.android.alarmclock/shared_prefs/AlarmClock.xml
-rw-rw----    1 10000    10000        4096 Dec 19  2008 /data/data/com.android.alarmclock/databases/alarms.db
-rw-rw----    1 10001    10001        7168 Nov 12 18:43 /data/data/org.koxx.forecast_weather.v2/databases/forecasts.d
b
-rw-rw----    1 10002    10002         489 Nov 11 23:19 /data/data/com.android.calculator2/files/calculator.data
-rw-rw----    1 10003    10003         683 Jun 10 19:27 /data/data/com.android.camera/shared_prefs/com.android.camera
_preferences.xml
-rw-rw----    1 10003    10003        5120 Dec 20  2008 /data/data/com.android.providers.drm/databases/drm.db
-rw-rw----    1 10003    10003       10240 Nov  1 16:24 /data/data/com.android.providers.downloads/databases/download
s.db
-rw-rw----    1 10003    10003       37888 May 13  2009 /data/data/com.android.providers.media/databases/internal.db
-rw-rw----    1 10003    10003       37888 Sep  4 23:25 /data/data/com.android.camera/databases/launcher.db
-rw-rw----    1 10003    10003       60416 Nov 12 19:01 /data/data/com.android.providers.media/databases/external-396
36438.db
-rw-r--r--    1 10004    10004           0 Jun 12 01:13 /data/data/com.android.providers.im/databases/im.db-mj76B91FF
8
-rw-r--r--    1 10004    10004           0 Jun 12 04:05 /data/data/com.android.providers.im/databases/im.db-mj0AB1E39
C
...
-rw-rw----    1 10004    10004         105 Dec 18  2008 /data/data/com.android.providers.contacts/shared_prefs/owner-
info.xml
-rw-rw----    1 10004    10004         125 Nov 11 16:37 /data/data/com.android.contacts/shared_prefs/dialtacts.xml
-rw-rw----    1 10004    10004         126 Dec 19  2008 /data/data/com.android.contacts/shared_prefs/_has_set_default
_values.xml
-rw-rw----    1 10004    10004         146 Aug 28 16:02 /data/data/com.android.contacts/shared_prefs/com.android.cont
acts_preferences.xml
-rw-rw----    1 10004    10004         169 Nov  5 02:01 /data/data/com.android.launcher/shared_prefs/launcher.xml
-rw-rw----    1 10004    10004        4096 Jan 30  2009 /data/data/com.android.providers.userdictionary/databases/use
r_dict.db
-rw-rw----    1 10004    10004       20480 Oct 31 21:12 /data/data/com.android.launcher/databases/launcher.db
-rw-rw----    1 10004    10004       21504 Nov 12 18:45 /data/data/com.android.providers.im/databases/im.db
-rw-rw----    1 10004    10004      110592 Nov 12 02:08 /data/data/com.android.providers.contacts/databases/contacts.
db
-rw-------    1 10005    10005         270 Jun 13 03:36 /data/data/com.android.email/databases/0c180cf8-fb7b-4d3e-b99
4-4282611af63a.db_att/32
-rw-r--r--    1 10005    10005     1418240 Nov  5 02:01 /data/data/com.android.email/databases/0c180cf8-fb7b-4d3e-b99
4-4282611af63a.db
-rw-rw----    1 10005    10005        1866 Dec 20  2008 /data/data/com.android.email/shared_prefs/AndroidMail.Main.xm
l
-rw-rw----    1 10005    10005        6144 Sep  8 01:35 /data/data/com.android.email/databases/webviewCache.db
-rw-rw----    1 10005    10005       14336 May 14 17:58 /data/data/com.android.email/databases/webview.db
-rw-rw----    1 10006    10006         126 Dec 18  2008 /data/data/com.google.android.gm/shared_prefs/_has_set_defaul
t_values.xml
-rw-rw----    1 10006    10006         199 Jan 22  2009 /data/data/com.google.android.gm/shared_prefs/Gmail.xml
-rw-rw----    1 10006    10006        6144 Dec 19  2008 /data/data/com.google.android.gm/databases/gmail.db
-rw-rw----    1 10006    10006        6144 Dec 23  2008 /data/data/com.google.android.gm/databases/webviewCache.db
-rw-rw----    1 10006    10006       14336 Dec 23  2008 /data/data/com.google.android.gm/databases/webview.db
-rw-------    1 10007    10007        1888 Nov 12 17:09 /data/data/com.google.android.apps.gtalkservice/files/sslcach
```

```
e/mtalk.google.com.5228
-rw-------    1 10007    10007        1954 Nov 12 18:43 /data/data/com.google.android.providers.gmail/files/sslcache/
android.clients.google.com.443
-rw-rw----    1 10007    10007        6144 Oct 23 22:43 /data/data/com.google.android.googleapps/databases/webviewCac
he.db
-rw-rw----    1 10007    10007        7168 May 13  2009 /data/data/com.google.android.providers.settings/databases/go
oglesettings.db
-rw-rw----    1 10007    10007       13312 Nov 11 20:37 /data/data/com.google.android.googleapps/databases/accounts.d
b
-rw-rw----    1 10007    10007       14336 May 13  2009 /data/data/com.google.android.googleapps/databases/webview.db
-rw-rw----    1 10007    10007      502784 Nov 12 18:45 /data/data/com.google.android.providers.gmail/databases/mails
tore.tbird20d@gmail.com.db
-rw-rw----    1 10009    10009         126 Sep  3  2008 /data/data/com.android.mms/shared_prefs/_has_set_default_valu
es.xml
-rw-rw----    1 10009    10009         585 Sep  3  2008 /data/data/com.android.mms/shared_prefs/com.android.mms_prefe
rences.xml
-rw-rw-rw-    1 10010    10010         310 Sep 18 01:12 /data/data/com.android.music/shared_prefs/Music.xml
-rw-rw----    1 10015    10015         126 Apr 29  2009 /data/data/com.google.android.street/shared_prefs/com.google.
android.street.StreetView.xml
-rw-------    1 10017    10017          35 Nov 12 16:49 /data/data/com.android.browser/cache/webviewCache/c24b0576
-rw-------    1 10017    10017          43 Nov 12 16:47 /data/data/com.android.browser/cache/webviewCache/5446c8f2
...
-rw-------    1 10017    10017     1204872 May 13  2009 /data/data/com.android.browser/app_plugins/gears.so
-rw-r--r--    1 10017    10017         512 Nov 12 19:18 /data/data/com.android.browser/databases/webviewCache.db-jour
nal
-rw-r--r--    1 10017    10017        8192 May 14 19:15 /data/data/com.android.browser/gears/geolocation.db
-rw-r--r--    1 10017    10017       18432 Dec 19  2008 /data/data/com.android.browser/gears/localserver.db
-rw-r--r--    1 10017    10017       20480 Dec 19  2008 /data/data/com.android.browser/gears/permissions.db
-rw-r--r--    1 10017    10017       48128 Nov 12 19:01 /data/data/com.android.browser/app_icons/WebpageIcons.db
-rw-rw----    1 10017    10017         851 May 29 13:53 /data/data/com.android.browser/shared_prefs/com.android.brows
er_preferences.xml
-rw-rw----    1 10017    10017       32768 Nov 12 16:49 /data/data/com.android.browser/databases/webviewCache.db
-rw-rw----    1 10017    10017       68608 Nov 12 16:49 /data/data/com.android.browser/databases/browser.db
-rw-rw----    1 10017    10017      257024 Nov 12 17:09 /data/data/com.android.browser/databases/webview.db
-rw-rw-rw-    1 10017    10017           0 Nov 12 16:48 /data/data/com.android.browser/app_plugins/gears-0.5.17.0/gea
rstimestamp
-rw-rw----    1 10018    10018         126 Sep  3  2008 /data/data/com.android.calendar/shared_prefs/_has_set_default
_values.xml
-rw-rw----    1 10018    10018         539 Nov 11 23:19 /data/data/com.android.calendar/shared_prefs/com.android.cale
ndar_preferences.xml
-rw-rw----    1 10018    10018      375808 Nov 12 09:58 /data/data/com.android.providers.calendar/databases/calendar.
db
-rw-rw----    1 10019    10019          48 Nov  7 06:11 /data/data/com.google.android.apps.maps/files/DATA_Tiles
-rw-rw----    1 10019    10019         483 Nov 11 03:58 /data/data/com.google.android.apps.maps/shared_prefs/com.goog
le.android.maps.MapsActivity.xml
-rw-rw----    1 10019    10019         708 Nov 12 17:09 /data/data/com.google.android.apps.maps/shared_prefs/friend_f
inder.xml
-rw-rw----    1 10019    10019        2000 Nov 11 03:58 /data/data/com.google.android.apps.maps/files/DATA_Preference
s
-rw-rw----    1 10019    10019        6144 May 13  2009 /data/data/com.google.android.apps.maps/databases/webviewCach
e.db
-rw-rw----    1 10019    10019        6144 Nov  1 21:28 /data/data/com.google.android.apps.maps/databases/search_hist
ory.db
-rw-rw----    1 10019    10019        8192 Nov 11 03:52 /data/data/com.google.android.apps.maps/databases/friends.db
-rw-rw----    1 10019    10019       14336 May 13  2009 /data/data/com.google.android.apps.maps/databases/webview.db
-rw-rw----    1 10019    10019       16048 Nov  7 06:11 /data/data/com.google.android.apps.maps/files/DATA_Tiles_1
-rw-rw-rw-    1 10019    10019          65 Nov 11 03:52 /data/data/com.google.android.apps.maps/shared_prefs/extra-fe
atures.xml
-rw-rw----    1 10021    10021         435 Oct 30 17:09 /data/data/com.android.vending/shared_prefs/vending_preferenc
es.xml
-rw-rw----    1 10021    10021        5120 Oct  6 19:38 /data/data/com.android.vending/databases/suggestions.db
-rw-rw----    1 10021    10021        6144 May 14 17:17 /data/data/com.android.vending/databases/webviewCache.db
-rw-rw----    1 10021    10021       14336 May 14 17:21 /data/data/com.android.vending/databases/webview.db
-rw-rw----    1 10021    10021       17408 Oct  6 19:39 /data/data/com.android.vending/databases/assets.db
-rw-------    1 10022    10022       50077 Nov 12 05:34 /data/data/com.google.android.youtube/cache/videos?vq=peter+s
ellers+inspector&format=2&restriction=us&start-index=18&max-results=8
-rw-------    1 10022    10022       53110 Nov 12 05:33 /data/data/com.google.android.youtube/cache/videos?vq=peter+s
ellers+inspector&format=2&restriction=us&start-index=10&max-results=8
-rw-------    1 10022    10022       57403 Nov 12 05:33 /data/data/com.google.android.youtube/cache/videos?vq=peter+s
ellers+inspector&format=2&restriction=us&start-index=1&max-results=9
-rw-------    1 10022    10022       63761 Nov 12 05:32 /data/data/com.google.android.youtube/cache/recently_featured
?format=2&start-index=1&max-results=9
```

```
-rw-rw----    1 10022    10022         739 Nov 12 16:45 /data/data/com.google.android.youtube/shared_prefs/youtube.xm
l
-rw-rw----    1 10022    10022        5120 Nov 12 05:34 /data/data/com.google.android.youtube/databases/suggestions.d
b
-rw-rw----    1 10025    10025         114 May 13  2009 /data/data/com.google.android.voicesearch/shared_prefs/com.go
ogle.android.voicesearch_preferences.xml
-rw-rw----    1 10025    10025         126 May 13  2009 /data/data/com.google.android.voicesearch/shared_prefs/_has_s
et_default_values.xml
-rw-rw----    1 10025    10025        2000 May 13  2009 /data/data/com.google.android.voicesearch/files/DATA_Preferen
ces
-rw-rw----    1 10025    10025        8192 Jun 10 03:25 /data/data/com.google.android.voicesearch/databases/webviewCa
che.db
-rw-rw----    1 10025    10025       14336 May 13  2009 /data/data/com.google.android.voicesearch/databases/webview.d
b
-rw-rw----    1 10026    10026         688 Jan 10  2009 /data/data/com.quirkconsulting/shared_prefs/TouchTipv2.xml
-rw-rw----    1 10026    10026        6144 Dec 19  2008 /data/data/com.quirkconsulting/databases/webviewCache.db
-rw-rw----    1 10026    10026       14336 Dec 19  2008 /data/data/com.quirkconsulting/databases/webview.db
-rw-------    1 10027    10027        4326 Aug  8 01:03 /data/data/com.a0soft.gphone.aCurrency/app_db/currency.db
-rw-rw----    1 10027    10027         170 Aug  8 01:04 /data/data/com.a0soft.gphone.aCurrency/shared_prefs/com.a0sof
t.gphone.aCurrency_preferences.xml
-rw-rw----    1 10027    10027         740 Dec 20  2008 /data/data/com.a0soft.gphone.aCurrency/shared_prefs/pref2.xml
-rw-rw----    1 10027    10027         801 Aug  8 01:05 /data/data/com.a0soft.gphone.aCurrency/shared_prefs/pref3.xml
-rw-rw----    1 10027    10027        6144 Aug  8 01:03 /data/data/com.a0soft.gphone.aCurrency/databases/webviewCache
.db
-rw-rw----    1 10027    10027       14336 Aug  8 01:03 /data/data/com.a0soft.gphone.aCurrency/databases/webview.db
-rw-rw----    1 10028    10028       14336 Sep 19 22:17 /data/data/com.stylem.movies/databases/webview.db
-rw-rw----    1 10028    10028       53248 Sep 19 22:18 /data/data/com.stylem.movies/databases/webviewCache.db
-rw-rw----    1 10029    10029         241 Jan 10  2009 /data/data/com.capaci.android.flashlight/shared_prefs/Setting
sFile.xml
-rw-rw----    1 10030    10030         233 Jun 15 15:02 /data/data/com.weather.Weather/files/tile-Radar-023010230-200
906151450-twc.png
-rw-rw----    1 10030    10030         233 Jun 15 15:02 /data/data/com.weather.Weather/files/tile-Radar-023010231-200
906151450-twc.png
-rw-rw----    1 10030    10030         233 Jun 15 15:02 /data/data/com.weather.Weather/files/tile-Radar-023010232-200
906151450-twc.png
-rw-rw----    1 10030    10030         233 Mar 18  2009 /data/data/com.weather.Weather/files/tile-Radar-023010221-200
903181940-twc.png
-rw-rw----    1 10030    10030         233 Mar 18  2009 /data/data/com.weather.Weather/files/tile-Radar-023010223-200
903181940-twc.png
-rw-rw----    1 10030    10030         233 Mar 18  2009 /data/data/com.weather.Weather/files/tile-Radar-023010230-200
903181940-twc.png
-rw-rw----    1 10030    10030         233 Mar 18  2009 /data/data/com.weather.Weather/files/tile-Radar-023010232-200
903181940-twc.png
```

When Dalvik (actually, the 'zygote' process') loads an application, it changes to the uid and gid for the application, so that the process is running in the correct security context.

Sample Process List

If you compare the Uids below with the above list, you'll see the correspondence between the assigned UID and the running processes. (For example, 10017 is the browser).

```
# ps
  PID  Uid        VSZ Stat Command
    1 0          288 S   /init
[kernel threads omitted...]
   30 1000        808 S   /system/bin/servicemanager
   31 0           848 S   /system/bin/vold
   32 0           668 S   /system/bin/debuggerd
   33 1001       7888 S   /system/bin/rild
   34 0         70548 S   zygote /bin/app_process -Xzygote /system/bin --zygote
   35 1013      30032 S   /system/bin/mediaserver
   36 1002       1172 S   /system/bin/dbus-daemon --system --nofork
   37 0           816 S   /system/bin/installd
   39 0           744 S   /system/bin/sh /runme.sh
   40 1008       1304 S   /system/bin/akmd
   41 0          3340 S   /sbin/adbd
   64 1000     171284 S   system_server
  108 1001     122172 S   com.android.phone
  110 10004    129936 S   android.process.acore
  387 10004    101668 S   com.android.inputmethod.latin
 6721 10017    158272 S   com.android.browser
 6901 10019     96340 S   com.google.android.apps.maps
 7166 0           740 S   /system/bin/sh -
 7635 10007    123776 S   com.google.process.gapps
 7727 10000     91284 S   com.android.alarmclock
 7753 0           872 S   sleep 3
 7754 0          2104 R   ps
```

Paranoid network-ing

Android adds a "paranoid network" option to the Linux kernel, which restricts access to some networking features depending on the group of the calling process.

The list of groups that are allowed access to networking features is in the kernel source file: */include/linux/android_aids.h*

Here is the list:

#define	GID	Capability
AID_NET_BT_ADMIN	3001	Can create an RFCOMM, SCO, or L2CAPP Bluetooth socket
AID_NET_BT	3002	Can create a Bluetooth socket
AID_INET	3003	Can create IPv4 or IPv6 socket
AID_NET_RAW	3004	Can create certain kinds of IPv4 sockets??
AID_NET_ADMIN*	3005	Allow CAP_NET_ADMIN permissions for process

Note: * Added in Donut (not in original Android 1.0)

Application-level security

Android also uses a user-space level security system to regulate communication and interaction among applications and system components. This is described at: http://developer.android.com/guide/topics/security/security.html

Tutorial

See http://siis.cse.psu.edu/android-tutorial.html

(With an abridged version at: http://siis.cse.psu.edu/android_sec_tutorial.html)

Security Analysis

See a good analysis of Android security at:
http://www.isecpartners.com/files/iSEC_Android_Exploratory_Blackhat_2009.pdf

Security Investigations

TOMOYO Linux investigation

Presentation

- "Learning, Analyzing and Protecting Android with TOMOYO Linux (JLS2009)" (Japan Linux Symposium 2009, Oct. 2009)
- "TOMOYO Linux on Android" (Smartbook/Netbook/Mobile Application Conference Taipei 2009, Oct. 2009)
- "TOMOYO Linux on Android" (CELF Japan Technical Jamboree 28, Jun. 2009)
- other English slides on slideshare of TOMOYO Linux
- all English slides PDFs of TOMOYO Linux

Code

- tomoyo-android-jp - Project Hosting on Google Code

Contact

haradats@gmail.com

Security Tricks

Changing application security permissions after installation

Some applications request more permissions than they really need. You can alter the set of permissions granted to an application by editing /data/system/pacakges.xml.

lbcoder wrote this on the android-platform mailing list:

```
Go into /data/system/packages.xml and you can remove permission lines.
Immediately after saving the packages.xml, reboot the phone (otherwise
the file will get overwritten by the system again). The newly reduced
permissions will be read on boot.
```

Category:

- Android

> From: eLinux.org

Android Memory Usage

The memory of an Android system is managed by several different allocators, in several different pools.

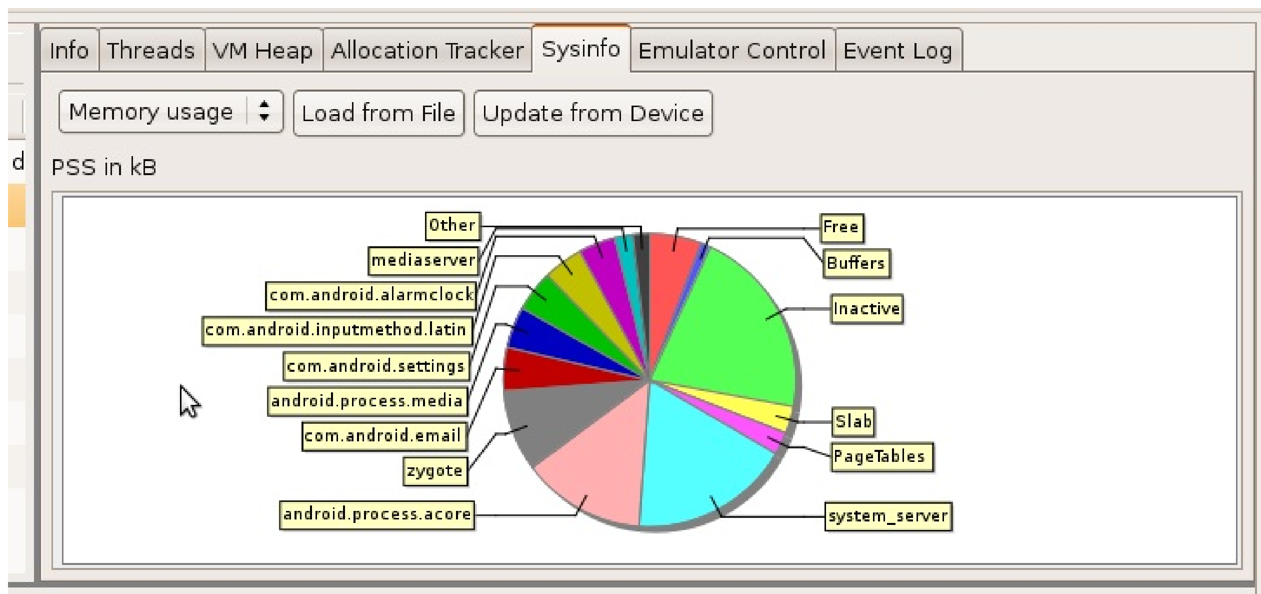
Contents

System Memory

You can examine the system's view of the memory on the machine, by examining /proc/meminfo.

If you use 'ddms', you can see a summary of the memory used on the machine, by the system and by the different executing processes. Click on the SysInfo tab, and select "Memory Usage" in the box on the upper left of the pane.

Here's a screenshot:



Note that you can get the numbers for each process by hovering your mouse over a particular pie slice. Numbers are shown in K and percentages.

Process Memory

You can see an individual process' memory usage by examining /proc/\/status

Details about memory usage are in

- /proc/\/statm
- /proc/\/maps
- /proc/\/smaps

The 'top' command will show VSS and RSS.

Also, see ddms info above.

procrank

procrank will show you a quick summary of process memory utilization. By default, it shows Vss, Rss, Pss and Uss, and sorts by Vss. However, you can control the sorting order.

procrank source is included in system/extras/procrank, and the binary is located in /system/xbin on an android device.

- Vss = virtual set size
- Rss = resident set size
- Pss = proportional set size
- Uss = unique set size

In general, the two numbers you want to watch are the Pss and Uss (Vss and Rss are generally worthless, because they don't accurately reflect a process's usage of pages shared with other processes.)

- Uss is the set of pages that are unique to a process. This is the amount of memory that would be freed if the application was terminated right now.
- Pss is the amount of memory shared with other processes, accounted in a way that the amount is divided evenly between the processes that share it. This is memory that would not be released if the process was terminated, but is indicative of the amount that this process is "contributing"

to the overall memory load.

You can also use procrank to view the working set size of each process, and to reset the working set size counters.

Here is procrank's usage:

```
# procrank -h
Usage: procrank [ -W ] [ -v | -r | -p | -u | -h ]
    -v  Sort by VSS.
    -r  Sort by RSS.
    -p  Sort by PSS.
    -u  Sort by USS.
        (Default sort order is PSS.)
    -R  Reverse sort order (default is descending).
    -w  Display statistics for working set only.
    -W  Reset working set of all processes.
    -h  Display this help screen.
```

And here is some sample output:

```
# procrank
  PID      Vss      Rss      Pss      Uss  cmdline
 1217   36848K   35648K   17983K   13956K  system_server
 1276   32200K   32200K   14048K   10116K  android.process.acore
 1189   26920K   26920K    9293K    5500K  zygote
 1321   20328K   20328K    4743K    2344K  android.process.media
 1356   20360K   20360K    4621K    2148K  com.android.email
 1303   20184K   20184K    4381K    1724K  com.android.settings
 1271   19888K   19888K    4297K    1764K  com.android.inputmethod.latin
 1332   19560K   19560K    3993K    1620K  com.android.alarmclock
 1187    5068K    5068K    2119K    1476K  /system/bin/mediaserver
 1384     436K     436K     248K     236K  procrank
    1     212K     212K     200K     200K  /init
  753     572K     572K     171K     136K  /system/bin/rild
  748     340K     340K     163K     152K  /system/bin/sh
  751     388K     388K     156K     140K  /system/bin/vold
 1215     148K     148K     136K     136K  /sbin/adbd
  757     352K     352K     117K      92K  /system/bin/dbus-daemon
  760     404K     404K     104K      80K  /system/bin/keystore
  759     312K     312K     102K      88K  /system/bin/installd
  749     288K     288K      96K      84K  /system/bin/servicemanager
  752     244K     244K      71K      60K  /system/bin/debuggerd
```

In this example, it shows that the native daemons and programs are an order of magnitude smaller than the Dalvik-based services and programs. Also, even the smallest Dalvik program requires about 1.5 meg (Uss) to run.

smem tool

You can see very detailed per-process or systemwide memory information with smem.

See Using smem on Android

Dalvik Heap

The Dalvik heap is preloaded with classes and data by zygote (loading over 1900 classes as of Android version 2.2). When zygote forks to start an android application, the new application gets a copy-on-write mapping of this heap. As Dan Borstein says below, this helps with memory reduction as well as application startup time.

Dalvik, like virtual machines for many other languages, does garbage collection on the heap. There appears to be a separate thread (called the HeapWorker) in each VM process that performs the garbage collection actions. (See toolbox ps -t) [need more notes on the garbage collection]

Dan Borstein said this about heap sharing[1]:

It's used in Android to amortize the RAM footprint of the large amount of effectively-read-only data (technically writable but rarely actually written) associated with common library classes across all active VM processes. 1000+ classes get preloaded by the system at boot time, and each class consumes at least a little heap for itself, including often pointing off to a constellation of other objects. The heap created by the preloading process gets shared copy-on-write with each spawned VM process (but again doesn't in practice get written much). This saves hundreds of kB of dirty unpageable RAM per process and also helps speed up process startup.

[INFO NEEDED: how to show dalvik heap info?]

Debugging Android application memory usage

See an excellent article by Dianne Hackborn at: http://stackoverflow.com/questions/2298208/how-to-discover-memory-usage-of-my-application-in-android/2299813#2299813

How to debug native process memory allocations

```
setprop dalvik.vm.checkjni true
setprop libc.debug.malloc 10
setprop setprop dalvik.vm.jniopts forcecopy
start
stop
```

libc.debug.malloc

The C library (bionic) in the system supports the ability to utilize a different, debug, version of the malloc code at runtime in the system.

If the system property libc.debug.malloc has a value other than 0, then when a process is instantiated, the C library uses functions for allocating and freeing memory, for that process.

(Note that there are other ways that the debug shared library malloc code ends up being used as well. That is, if you are running in the emulator, and the value of the system property ro.kernel.memcheck is not '0', then you get a debug level of 20. Note that debug level 20 can only be used in the emulator.)

By default, the standard malloc/free/calloc/realloc/memalign routines are used. By setting libc.debug.malloc, different routines are used, which check for certain kinds of memory errors (such as leaks and overruns). This is done by loading a separate shared library (.so) with these different routines.

The shared libraries are named: /system/lib/libc_malloc_debug_leak.so and /system/lib/libc_malloc_debug_qemu.so

(Information was obtained by looking at \/bionic/libc/bionic/malloc_debug_common.c)

Supported values for libc.debug.malloc (debug level values) are:

- 1 - perform leak detection
- 5 - fill allocated memory to detect overruns
- 10 - fill memory and add sentinels to detect overruns
- 20 - use special instrumented malloc/free routines for the emulator

I'm not sure whether these shared libraries are shipped in production devices.

References

1. ↑ comment by Dan Borstein, Jan 2009 to blog article Dalvik vs. Mono

Category:

- Android

> From: eLinux.org

Android Dalvik VM

Dalvik is the name of the Virtual Machine in which Android applications are run. This VM executes Dalvik bytecode, which is compiled from programs written in the Java language. Note that the Dalvik VM is not a Java VM (JVM).

Every Android application runs in its own process, with its own instance of the Dalvik virtual machine.

At boot time, a single virtual machine, called 'zygote' is created, which preloads a long list of classes. (As of Android version 2.1 (eclair), the list of classes preloaded by zygote had 1,942 entries). All other "java" programs or services are forked from this process, and run as their own process (and threads) in their own address space. Both applications and system services in the Android framework are implemented in "java".

Dalvik was written so that a device can run multiple VMs efficiently. The Dalvik VM executes code in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool.

Most Android applications are delivered and stored on the system as packages (.apks), which include both dex bytes code (classes and methods) and resources. During first boot-up the system creates a cache of dex classes in /data/dalvik-cache.

Contents

Documentation

There is some documentation on Dalvik in the source code in the dalvik/docs directory. See the Android dalvik docs git repository.

The source code has some rather large comments, including near the top of Thread.c and Exception.c. The "mterp" directory has some notes describing the structure of the interpreters.

Q&A

From fadden on the android-platform list:

- Who is responsible to read the dex file and call Dalvik ?
  - The VM is started by the framework, through the JNI invocation interface. See AndroidRuntime.cpp in frameworks/base/...
- Do you have a tutorial launch an app in the terminal adb: bytecode file > dex file > app launch in Dalvik => and see the result in adb ?
  - See "hello-world.html" in the dalvik/docs directory (linked above).
- Is there a profiler inside Dalvik, which enable us to follow each step in the dalvik execution ?

  - You could turn on LOG_INSTR to see each instruction as it is executed. This results in a rather dramatic amount of logging. If you try to do this on a device you will overrun the 64KB kernel log buffer pretty quickly and drop lots of stuff, so it's really only suitable for a "desktop" build (e.g. sim-eng).
  - Also, Dalvik does include instrumentation to allow for tracing and profiling. See http://elinux.org/Android_Tools#traceview

JIT

As of version 2.2 (*Froyo*), Dalvik includes a Just-In-Time compiler (or JIT).

- See A JIT Compiler for Android's Dalvik VM
  - video of presentation by Ben Cheng and Bill Buzbee at Google IO, 2010
  - Slides, in PDF

The Dalvik JIT, as of version 2.2, is a "trace-granularity JIT", which means that it compiles individual code fragments that it discovers at runtime to be "hot spots". (That is, it does not compile whole methods.) The Dalvik bytecode interpreter is constantly profiling the code it is executing, and when a piece of code is determined to be running a lot, it is passed to a compiler to turn into native code. Several optimizations may be performed in this process. This code is then executed instead of the bytecode, for future runs through this section of the software.

The memory overhead of the JIT is reported to be between 100K to 200K per application. The ratio of code size between native instructions and DEX byte codes in one example give (see slide 22 of the presentation) was 7.7 to 1. That is, native instructions take approximately 8 times as much space as DEX byte codes do to perform the same operations.

Relationship to Java

Because Dalvik is not referred to as a Java Virtual Machine it does not utilize the branding of "Java". Also, it does not execute Java bytecodes. Hence, Google can ignore licensing issues with Sun or Oracle, with regards to Java.

However, a Java compiler and set of class libraries are required in order to create a Dalvik program.

As of March 2010, only the Sun JDK, version 1.5 is supported for building the Android system and add-on Android applications.

Dalvik on other platforms

- Myriad Alien Dalvik - an implementation of the Dalvik VM for other platforms (demonstrated first on Meego)
  - http://www.linuxfordevices.com/c/a/News/Myriad-Group-Myriad-Alien-Dalvik/?kc=LNXDEVNL020911

Debugging the VM

There are a number of properties you can set, to control operation of the VM and allow for debugging various aspects of the system:

See http://netmite.com/android/mydroid/dalvik/docs/embedded-vm-control.html

(Note that this is in \/dalvik/docs, along with a whole bunch of other files with information about Dalvik.)

this mentions a number of features you can control with properties, including:

- checkjni - various checks when JNI is used
- enableassertions - enable assertions in the VM code
- verify-bytecode - whether to perform bytecode verification
- execution-mode - whether to use optimized assembly or portable C code for the interpreter
- stack-trace-file - where to put stack trace data when a SIGQUIT is received

Getting stack traces

You can force the VM to dump a stack trace of all threads by sending a SIGQUIT signal. This can be done using 'kill -3 \', where pid in the main dalvik process. By default, the stack trace goes to the android log, but you can have the data sent to a file (using the dalvik.vm.stack-trace-file property) instead.

Using checkJNI

CheckJNI refers to a special runtime mode of the Dalvik VM, which forces the VM to run certain check and report problems when it sees certain errors occuring from code called via JNI.

See http://android-developers.blogspot.com/2011/07/debugging-android-jni-with-checkjni.html

Resources

- Dalvik wikipedia entry)
- Dalvik VM Internals - video of presentation by Dan Bornstein at Google IO, 2008
- DEX file format, reverse engineered by Michael Pavone
- Dalvik bytecodes

Category:

- Android

> From: eLinux.org

Android Packages

Contents

Packages

Android applications are shipped as "packages", which are compressed archives with class files, resources, and meta-information (the AndroidManifest.xml file and security certificates) for the application.

A package has the extension .apk, and it consists of a set of directories and files in a gzip'ed archive.

A package usually contains the following items:

META-INF/MANIFEST.MF The manifest file for the package file (apk) itself.

META-INF/CERT.SF A security certificate

META-INF/CERT.RSA another security certificate (should distinguish these two)

AndroidManifest.xml The manifest file for the application(s) in this package

classes.dex The actual code for the dalvik (http://eLinux.org/java) classes for the application(s) in this package

res/ a directory containing resource files

resources.arsc ???

AndroidManifest.xml

The AndroidManifest.xml file has information about the application(s) in a package. Usually, a package will contain a single application. The AndroidManifest file describes the application's name, as well as libraries and security permissions neede by the app, messages used by the app, what icon to use to represent the app, and more.

See http://developer.android.com/guide/topics/manifest/manifest-intro.html for details.

Tools for managing packages

The aapt tool is used to create, inspect and modify Android packages.

Resources

An overview of application resources is at: http://developer.android.com/guide/topics/resources/index.html

It is possible to use a raw file as a resource (without it getting compiled by the build system). See this article on using raw files as resources in Android.

Assets

Assets are like resources, except that do not have a resource ID, and they are listed in the 'assets' directory of a package, rather than the 'res' directory.

Category:

- Android

> From: eLinux.org

Android Networking

Here are some tips for getting networking working on your Android device.

Setting the DNS server

In my experience, some times the device fails to get it's DNS server correctly from DHCP. The DNS server address is retrieved from system properties.

You can set this manually by typing (on the target):

```
$ setprop net.eth0.dns1 xx.yy.zz.aa
```

(replacing with the correct numeric values for your DNS server address)

I have also set the following:

```
$ setprop net.nds1 xx.yy.zz.aa
```

This needs to be set each time the system boots. I'm sure there's a way to have 'init' set this, or to put it into persistent properties, but I haven't done that yet.

Configuring a web proxy

The browser looks at system settings stored in a provider database for the http proxy.

The value for http_proxy is set in the following sqlite database:
/data/data/com.android.providers.settings/databases/settings.db

the setting goes in the 'system' table in this database, with a key of 99, a name of 'http_proxy' and a value that is a string containing your proxy server and port. I don't know if you can use a server name rather than just an IP address. I haven't done that.

To set this, on target, do the following:

```
 # cd /data/data/com.android.providers.settings/databases
 # sqlite3 settings.db
SQLite version 3.5.9
Enter ".help" for instructions
sqlite> insert into system values(99,'http_proxy','192.168.1.1:80');
sqlite>.exit
```

Replace '192.168.1.1:80' with the appropriate address and port for your network configuration.

Since it's only a single line, you can also do this from the command line. You can check that the value is stored properly by typing the following:

```
# sqlite3 settings.db "select * from system;"
```

This should show output similar to the following (first part omitted):

```
...
38|volume_ring|4
39|volume_ring_last_audible|4
99|http_proxy|43.131.3.73:80
```

This change is persistent, and you should only have to make it once.

Setting up networking on bootup

Add the following lines in the init.rc:

```
service ethernet /eth0.sh
```

Now create a file at the root (or anywhere else, probably in /system/etc but modify the above line accordingly) called eth0.sh, make it executable (chmod +x eth0.sh) and put the following lines in it:

```
#!/system/bin/sh

netcfg eth0 dhcp

setprop net.dns1 8.8.8.8
setprop net.dns2 8.8.4.4
```

Category:

- Android

> From: eLinux.org

Android File Systems

YAFFS

Android in most mobile phones up to version 2.2 (Froyo) use the YAFFS2 filesystem.

Here is some information about YAFFS2:

- File Systems#YAFFS2
- Wookey's presentation about YAFFS at ELC Europe 2007 yaffs.pdf, video
- YAFFS update for ELC Europe 2010

EXT4

According to this report by Ted Tso, version 2.3 (Gingerbread) will be using ext4

http://www.linuxfoundation.org/news-media/blogs/browse/2010/12/android-will-be-using-ext4-starting-gingerbread

With ext4, you may need to explicitly sync the data to the file system, in order to make sure not to lose information.

See this blog entry by Tim Bray for information on this topic.

Category:

- Android

> From: eLinux.org

Android Logging System

This article describes the Android logging system

Contents

Overview

The Android system has a logging facility that allows systemwide logging of information, from applications and system components. This is separate from the Linux kernel's own logging system, which is accessed using 'dmesg' or '/proc/kmsg'. However, the logging system does store messages in kernel buffers.

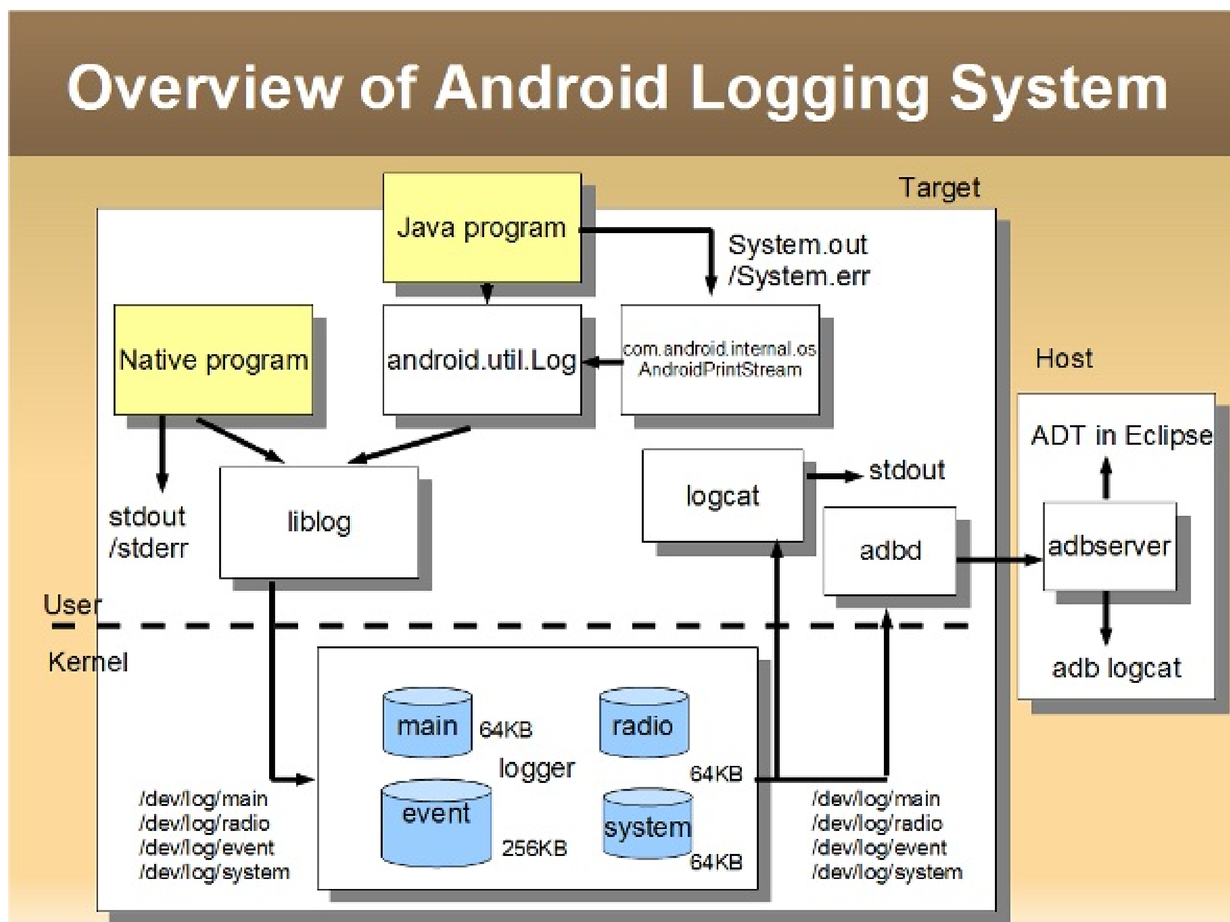


image by Tetsuyuki Kobabayshi, of Kyoto Microcomputer Co.

The logging system consists of:

- a kernel driver and kernel buffers for storing log messages
- C, C++ and Java classes for making log entries and for accessing the log messages
- a standalone program for viewing log messages (logcat)
- ability to view and filter the log messages from the host machine (via eclipse or ddms)

There are four different log buffers in the Linux kernel, which provide logging for different parts of the system. Access to the different buffers is via device nodes in the file system, in /dev/log.

The four log buffers are:

- main - the main application log
- events - for system event information
- radio - for radio and phone-related informatio
- system - a log for low-level system messages and debugging

Up until 2010, only the first three logs existed. The system log was created to keep system messages in a separate buffer (outside of '/dev/log/main') so that a single verbose application couldn't overrun system messages and cause them to be lost.

Each message in the log consists of a tag indicating the part of the system or application that the message came from, a timestamp, the message log level (or *priority* of the event represented by the message) and the log message itself.

All of the log buffers except for 'event' use free-form text messages. The 'event' buffer is a 'binary' buffer, where the event messages (and event parameters) are stored in binary form. This form is more compact, but requires extra processing when the event is read from the buffer, as well as a message lookup database, to decode the event strings.

The logging system automatically routes messages with specific tags into the radio buffer. Other messages are placed into their respective buffers when the the log class or library for that buffer is used.

Kernel driver

The kernel driver for logging is called the 'logger'. See Android logger

System and Application logging

Application log

An Android application includes the android.util.Log class, and uses methods of this class to write messages of different priority into the log.

Java classes declare their tag statically as a string, which they pass to the log method. The log method used indicates the message "severity" (or log level). Messages can be filtered by tag or priority when the logs are processed by retrieval tools (logcat).

Event log

Event logs messages are created using android.util.EventLog class, which create binary-formatted log messages. Log entries consist of binary tag codes, followed by binary parameters. The message tag codes are stored on the system at: /system/etc/event-log-tags. Each message has the string for the log message, as well as codes indicating the values associated with (stored with) that entry.

System log

Many classes in the Android framework utilize the system log to keep their messages separate from (possibly noisy) application log messages. These programs use the android.util.Slog class, with its associated messages.

In all cases, eventually a formatted message is delivered through the C/C++ library down to the kernel driver, which stores the message in the appropriate buffer.

'log' command line tool

There is a 'log' command line tool that can be used to create log entries from any program. This is built into the 'toolbox' multi-function program.

The usage for this is:

```
USAGE: log [-p priorityChar] [-t tag] message
       priorityChar should be one of:
               v,d,i,w,e
```

Capturing stdout with logwrapper

It is sometimes useful to capture stdout from native applications into the log. There is a utility called 'logwrapper' which can be used to run a program, and redirect it's stdout into log messages.

The logwrapper usage is:

```
Usage: logwrapper [-x] BINARY [ARGS ...]

Forks and executes BINARY ARGS, redirecting stdout and stderr to
the Android logging system. Tag is set to BINARY, priority is
always LOG_INFO.

-x: Causes logwrapper to SIGSEGV when BINARY terminates
    fault address is set to the status of wait()
```

Source for logwrapper is at: system/core/logwrapper/logwrapper.c

Logcat command

You can use the 'logcat' command to read the log. This command is located in /system/bin in the local filesystem, or you can access the functionality using the 'adb logcat' command.

Documentation on the use of this command is at: http://developer.android.com/guide/developing/tools/adb.html

Some quick notes:

- Log messages each have a tag and priority.
    - You can filter the messages by tag and log level, with a different level per tag.
- You can specify (using a system property) that various programs emit their stdout or stderr to the log.

Trick to couple Android logging to Linux kernel logging

Note that the android logging system is completely separate from the Linux kernel log system (which uses printk inside the kernel to save messages and dmesg to extract them). You can write to the kernel log from user space by writing to /dev/kmsg.

I've seen a reference to couple the two together, to redirect android messages into the kernel log buffer, using the 'logcat' program launched from init, like below:

```
service logcat /system/bin/logcat -f /dev/kmsg
      oneshot
```

(See http://groups.google.com/group/android-kernel/browse_thread/thread/87d929863ce7c29e/f8b0da9ed6376b2f?pli=1)

I'm not sure why you'd want to do this (maybe to put all messages into a single stream? With the same timestamp, if kernel message timestamping is on?)

Resources

- Android Debug Bridge reference page
  - has 'adb logcat' usage information
- Logging System Of Android Presentation by Tetsuyuki Kobayashi, September, 2010 at CELF's Japan Technical Jamboree 34

Category:

- Android

> From: eLinux.org

Android Source Code Description

Android source code is massive and contains several projects and this is humanly impossible to remember each and every single projects available. This page has been brought to live to enable developers to understand the different part of the source code. This page contains different version of Android and it will outline in detail what each directory contains. Following are the links to the different Android version source code information

- android-4.0.3_r1
- android-4.1.1_r4
- master-android
- AOSP Contribution

Written and Maintained by : Nanik Tolaram

Category:

- Android

Software development

About Android software development, include SDK, build system, tools, debugging and testing.

> From: eLinux.org

Android SDK

Android Developers Homepage

Application SDK

Used for creating high level portable Java code.

1. Android SDK (Required)
2. Eclipse Classic or greater (Not technically required
3. Eclipse ADT Plugin (Required if using Eclipse)
4. Netbeans plugin (No official support)
5. Windows USB Driver (Only need if using Windows)

Versions Regularly Updated Dashboard

Top 3 SDK in used by users as of 4/13/2010

1. 1.5
2. 1.6
3. 2.0

Native SDK

Used to create a portable shared object library file for your C/C++ or asm code inside your Java application

1. Android NDK

Category:

- Android

> From: eLinux.org

Android Build System

Basics of the Android Build system were described at (**Down as of 9/2011**):
http://source.android.com/porting/build_system.html

Note that "partner-setup" should be replaced with "choosecombo" or even "lunch" in that description.

(*This information seems to be duplicated at:* http://pdk.android.com/online-pdk/guide/build_system.html I'm going to assume that the source.android site is the most up-to-date, but I haven't checked.)

More information about the Android build system, and some of the rationale for it, are described at:
http://android.git.kernel.org/?p=platform/build.git;a=blob_plain;f=core/build-system.html

You use build/envsetup.sh to set up a "convenience environment" for working on the Android source code. This file should be source'ed into your current shell environment. After doing so you can type 'help' (or 'hmm') for a list of defined functions which are helpful for interacting with the source.

Contents

Overview

The build system uses some pre-set environment variables and a series of 'make' files in order to build an Android system and prepare it for deployment to a platform.

Android make files end in the extension '.mk' by convention, with the *main* make file in any particular source directory being named 'Android.mk'.

There is only one official file named 'Makefile', at the top of the source tree for the whole repository. You set some environment variables, then type 'make' to build stuff. You can add some options to the make command line (other targets) to turn on verbose output, or perform different actions.

The build output is placed in 'out/host' and 'out/target' Stuff under 'out/host' are things compiled for your host platform (your desktop machine). Stuff under 'out/target/product/\' eventually makes it's way to a target device (or emulator).

The directory 'out/target/product/\/obj' is used for staging "object" files, which are intermediate binary images used for building the final programs. Stuff that actually lands in the file system of the target is stored in the directories root, system, and data, under 'out/target/product/\'. Usually, these are bundled up into image files called system.img, ramdisk.img, and userdata.img.

This matches the separate file system partitions used on most Android devices.

Some Details

What tools are used

During the build you will be using 'make' to control the build steps themselves. A host toolchain (compiler, linker and other tools) and libraries will be used to build programs and tools that will run on the host. A different toolchain is used to compile the C and C++ code that will wind up on the target (which is an embedded board, device or the emulator). This is usually a "cross" toolchain that runs on an X86 platform, but produces code for some other platform (most commonly ARM). The kernel is compiled as a standalone binary (it does not use a program loader or link to any outside libraries). Other items, like native programs (e.g. init or toolbox), daemons or libraries will link against bionic or other system libraries.

You will be using a Java compiler and a bunch of java-related tools to build most of the application framework, system services and Android applications themselves. Finally, tools are used to package the applications and resource files, and to create the filesystem images that can be installed on a device or used with the simulator.

Telling the system where the Java toolchain is

Before you build anything, you have to tell the Android build system where your Java SDK is. (Installing a Java SDK is a pre-requisite for building).

Do this by setting a JAVA_HOME environment variable.

Specifying what to build

In order to decide what to build, and how to build it, the build system requires that some variables be set. Different products, with different packages and options can be built from the same source tree. The variables to control this can be set via a file with declarations of 'make' variables, or can be specified in the environment.

A device vendor can create definition files that describe what is to be included on a particular board or for a particular product. The definition file is called: buildspec.mk, and it is located in the top-level source directory. You can edit this manually to hardcode your selections.

If you have a buildspec.mk file, it sets all the make variables needed for a build, and you don't have to mess with options.

Another method of specifying options is to set environment variables. The build system has a rather ornate method of managing these options for you.

To set up your build environment, you need to load the variables and functions in build/envsetup.sh. Do this by 'source-ing' the file into your shell environment, like this:

```
$ source build/envsetup.sh
```

or

```
$ . build/envsetup.sh
```

You can type 'help' (or 'hmm') at this point to see some utility functions that are available to make it easier to work with the source.

To select the set of things you want to build, and what items to build for, you use either the 'choosecombo' function or the 'lunch' function. 'choosecombo' will walk you through the different items you have to select, one-by-one, while 'lunch' allows you select some pre-set combinations.

The items that have to be defined for a build are:

- the product ('generic' or some specific board or platform name)
- the build variant ('user', 'userdebug', or 'eng')
- whether you're running on a simulator ('true' or 'false')
- the build type ('release' or 'debug')

Descriptions of these different build variants are at
http://source.android.com/porting/build_system.html#androidBuildVariants

The build process from a user perspective is described pretty well in this blog post: First Android platform build by CodePainters, December 2009

Actually building the system

Once you have things set up, you actually build the system with the 'make' command.

To build the whole thing, run 'make' in the top directory. The build will take a long time, if you are building everything (for example, the first time you do it).

Build tricks

Seeing the actual commands used to build the software

Use the "showcommands" target on your 'make' line:

```
$ make -j4 showcommands
```

This can be used in conjunction with another make target, to see the commands for that build. That is, 'showcommands' is not a target itself, but just a modifier for the specified build.

In the example above, the -j4 is unrelated to the showcommands option, and is used to execute 4 make sessions that run in parallel.

Make targets

Here is a list of different make targets you can use to build different parts of the system:

- make sdk - build the tools that are part of an SDK (adb, fastboot, etc.)
- make snod - build the system image from the current software binaries
- make services
- make runtime
- make droid - make droid is the normal build.
- make all - make everything, whether it is included in the product definition or not
- make clean - remove all built files (prepare for a new build). Same as rm -rf out/\/
- make modules - shows a list of submodules that can be built (List of all LOCAL_MODULE definitions)
- make \ - make a specific module (note that this is not the same as directory name. It is the LOCAL_MODULE definition in the Android.mk file)
- make clean-\ - clean a specific module
- make bootimage TARGET_PREBUILT_KERNEL=/path/to/bzImage - create a new boot image with custom bzImage

Helper macros and functions

There are some helper macros and functions that are installed when you source envsetup.sh. They are documented at the top of envesetup.sh, but here is information about a few of them:

- croot - change directory to the top of the tree
- m - execute 'make' from the top of the tree (even if your current directory is somewhere else)
- mm - builds all of the modules in the current directory
- mmm \ ... - build all of the modules in the supplied directories
- cgrep \ - grep on all local C/C++ files
- jgrep \ - grep on all local Java files
- resgrep \ - grep on all local res/*.xml files
- godir \ - go to the directory containing a file

Speeding up the build

You can use the '-j' option with make, to start multiple threads of make execution concurrently.

In my experience, you should specify about 2 more threads than you have processors on your machine. If you have 2 processors, use 'make -j4', If they are hyperthreaded (meaning you have 4 virtual processors), try 'make -j6.

You can also specify to use the 'ccache' compiler cache, which will speed up things once you have built things a first time. To do this, specify 'export USE_CCACHE=1' at your shell command line. (Note that ccache is included in the prebuilt section of the repository, and does not have to be installed on your host separately.)

Building only an individual program or module

If you use build/envsetup.sh, you can use some of the defined functions to build only a part of the tree. Use the 'mm' or 'mmm' commands to do this.

The 'mm' command makes stuff in the current directory (and sub-directories, I believe). With the 'mmm' command, you specify a directory or list of directories, and it builds those.

To install your changes, do 'make snod' from the top of tree. 'make snod' builds a new system image from current binaries.

Setting module-specific build parameters

Some code in Android system can be customized in the way they are built (separate from the build variant and release vs. debug options). You can set variables that control individual build options, either by setting them in the environment or by passing them directly to 'make' (or the 'm...' functions which call 'make'.)

For example, the 'init' program can be built with support for bootchart logging by setting the INIT_BOOTCHART variable. (See Using Bootchart on Android for why you might want to do this.)

You can accomplish either with:

```
$ touch system/init/init.c
$ export INIT_BOOTCHART=true
$ make
```

or

```
$ touch system/init/init.c
$ m INIT_BOOTCHART=true
```

Makefile tricks

These are some tips for things you can use in your own Android.mk files.

build helper functions

A whole bunch of build helper functions are defined in the file build/core/definitions.mk

Try grep define build/core/definitions.mk for an exhaustive list.

Here are some possibly interesting functions:

- print-vars - shall all Makefile variables, for debugging
- emit-line - output a line during building, to a file
- dump-words-to-file - output a list of words to a file
- copy-one-file - copy a file from one place to another (dest on target?)

Add a file directly to the output area

You can copy a file directly to the output area, without building anything, using the add-prebuilt-files function.

The following line, extracted from prebuilt/android-arm/gdbserver/Android.mk copies a list of files to the EXECUTABLES directory in the output area:

```
$(call add-prebuilt-files, EXECUTABLES, $(prebuilt_files))
```

Adding a new program to build

Steps for adding a new program to the Android source tree

- make a directory under 'external'
  - e.g. ANDROID/external/myprogram
- create your C/cpp files.
- create Android.mk as clone of external/ping/Android.mk
- Change the names ping.c and ping to match your C/cpp files and program name
- add the directory name in ANDROID/build/core/main.mk after external/zlib as external/myprogram
- make from the root of the source tree
- your files will show up in the build output area, and in system images.
  - You can copy your file from the build output area, under out/target/product/..., if you want to copy it individually to the target (not do a whole install)

See http://www.aton.com/android-native-development-using-the-android-open-source-project/ for a lot more detail.

Building the kernel

The kernel is "outside" of the normal Android build system (indeed, the kernel is not included by default in the Android Open Source Project). However, there are tools in AOSP for building a kernel. If you are building the kernel, start on this page: http://source.android.com/source/building-kernels.html

If you are building the kernel for the emulator, you may also want to look at: http://stackoverflow.com/questions/1809774/android-kernel-compile-and-test-with-android-emulator

And, Ron M wrote (on the android-kernel mailing list, on May 21, 2012):

This post is very old - but nothing has changes as far as AOSP is concerned, so in case anyone is interested and runs into this problem when building for QEMU:

There is actually a nice and shorter way to build the kernel for your QEMU target provided by the AOSP:

1. cd to your kernel source dir (Only goldfish 2.6.29 works out of the box for the emulator)
2. ${ANDROID_BUILD_TOP}/external/qemu/distrib/build-kernel.sh -j=64 --arch=x86 --out=$YourOutDir
3. emulator -kernel ${YourOutDir}/kernel-qemu # run emulator:

Step #2 calls the toolbox.sh wrapper scripts which works around the SSE disabling gcc warning - which happens for GCC \ < 4.5 (as in the AOSP prebuilt X86 toolchain).

This script adds the " -mfpmath=387 -fno-pic" in case it is an X86 and that in turn eliminates the compilation errors seen above.

To have finer control over the build process, you can use the "toolbox.sh" wrapper and set some other stuff without modifying the script files.

An example for building the same emulator is below:

```
# Set arch
export ARCH=x86
# Have make refer to the QEMU wrapper script for building android over x86
(eliminates the errors listed above)
export
CROSS_COMPILE=${ANDROID_BUILD_TOP}/external/qemu/distrib/kernel-toolchain/android-kernel-toolchain-
# Put your cross compiler here. I am using the AOSP prebuilt one in this example
export
REAL_CROSS_COMPILE=${ANDROID_BUILD_TOP}/prebuilt/linux-x86/toolchain/i686-android-linux-4.4.3/bin/i686-android-linux-
# Configure your kernel - here I am taking the default goldfish_defconfig
make goldfish_defconfig
# build
make -j64
# Run emulator:
emulator -kernel arch/x86/boot/bzImage -show-kernel
```

This works for the 2.6.29 goldfish branch. If anyone is using the emulator with a 3+ kernel I would be like to hear about it.

Category:

- Android

> From: eLinux.org

Android Application Development

For the most reliable reference to the Android Application Development go through:

http://developer.android.com/guide/index.html

This includes java doc to all classes, packages, and other variables and is the most updated site.

Though it is most customary to get started in Eclipse, which is the most standard, you are also free to use Netbeans, but the Netbeans updates for the Android SDK plugin is not very dynamic and to my knowledge was developed and supported by a single project called Kenai.

You should also know that you can not step into Android Application Development with out having base on Core Java programming, yes, that is a pre-requisite, else it is bit difficult.

So then, welcome, get ready to jump into the droid community and get going.

Category:

- Android

> From: eLinux.org

Android Scripting

SL4A (Scripting Layer for Android) allowing you to edit and execute scripts and interactive interpreters directly on the Android device. Many parts of the Android API are available, directly from the scripting language.

Scripts can be run interactively in a terminal or associated with an icon and launched like regular apps.

Home page: http://code.google.com/p/android-scripting/

The SL4A User's Guide is a good place to start.

SL4A supports the following languages

- Python
- Perl
- JRuby
- Lua
- BeanShell
- JavaScript
- Tcl
- Shell scripts

Category:

- Android

> From: eLinux.org

Android Debugging

Debugging methods for Android

Contents

Debuggers

Kernel and User co-debug with GDB on Android

This presentation covers lots of Android debug resources provided by Linaro, presented by Zach Pfeffer in Spring 2012. Included is information about how to debug kernel and user simultaneously with gdb.

See Media:Zach Pfeffer Next Gen Android 2012.pdf

loggers

kernel message log

The Linux kernel has a message log in an internal ring buffer. You can access the contents of this log using the 'dmesg' command.

You can add timing information to the printk messages, by adding "time" to the Linux kernel command line.

init logging

The Android init program outputs some messages to the kernel log, as it starts the system. You can increase the verbosity of init, using the "loglevel" command in the /init.rc file.

The default loglevel is 3, but you can change it to 8 (the highest) by changing the following line in the /init.rc file. Change:

```
loglevel 3
```

to

```
loglevel 8
```

Android logging system

The Android application framework has a built-in logging system, which goes through a special driver in the kernel. It is described at:http://developer.android.com/guide/developing/tools/adb.html#logcat

Note that although the log dumper (logcat) is described on the Android developer site on the 'adb' page, there is a native logcat command included in the Android distribution (that is, available on the target file system, which can be run locally).

tracers

strace

You can use strace on Android. It is included in the Android open source project (at least as of Android 2.1), and appears to be automatically installed in engineering builds of the software.

To use strace during early initialization, you can put it in the /init.rc file. For example, to trace zygote initialization, change the following line in /init.rc.

```
service zygote /system/bin/app_process -Xzygote /system/bin --zygote --start-system-server
```

should be changed to:

```
service zygote /system/xbin/strace -tt -o/data/boot.strace /system/bin/app_process -Xzygote /system/bin --zygote --st
art system-server
```

Dalvik Method Tracer

See http://developer.android.com/guide/developing/tools/traceview.html

Additional Resources

- Google I/O 2009 - Debugging Arts of the Ninja Masters (video) by Justin Mattson

Category:

- Android

> From: eLinux.org

Android Testing

This page has information about various tests you can perform on Android

Google provides a number of resources for testing applications, including monkey, the logger, and the compliance test suite (CTS)

For an overview of Android application testing and resources provided by Google, see: http://developer.android.com/guide/topics/testing/index.html

Contents

Android Test Framework

Google provides an integrated test framework for testing Android applications, based on the JUnit test framework from java.

See Android Testing Fundamentals

Application testing resources

- monkeyrunner
  - The monkeyrunner tool provides an API for writing programs that control an Android device or emulator from outside of Android code. It allows you to write a program, which runs on your host machine that can interact with an application running in the emulator or on a target device.
  - With monkeyrunner, you can write a Python program that installs an Android application or test package, runs it, sends keystrokes to it, takes screenshots of its user interface, and stores screenshots on the host.
- Monkey is a user interface and application tester for Android applications.
  - It is a command-line tool that sends pseudo-random streams of keystrokes, touches, and gestures to a device.
  - This tool in unrelated to the monkeyrunner tool mentioned above. (It runs on the target, and monkeyrunner-based programs run on the development host machine.)
- Robotium test framework
  - Robotium is a test framework created to make it easy to write powerful and robust automatic black-box test cases for Android applications. With the support of Robotium, test case developers can write function, system and acceptance test scenarios, spanning multiple Android activities.
  - Robotium has full support for Activities, Dialogs, Toasts, Menus and Context Menus.
- Roboelectric test framework
  - Roboelectric allow you to test-drive the development of your Android app inside the JVM on your workstation in seconds, instead of in the emulator on on a device (which can be slow)
  - Robolectric allows you to test most Android functionality including layouts and GUI behavior, services, and networking code. It has more flexibility than Google's testing framework in some areas.
- Ranorex test framework
  - Ranorex provides a .Net based test automation framework which allows to record test scenarios directly on real

mobile devices as well as on desktop machines.
  - Ranorex easily instruments the apk and deploys the instrumetned apk directly on the devices, starts the application automatically, performs all recorded actions (keystrokes, touch events, device button events, validations) and closes the application automatically.

Benchmarks

There are a number of benchmarks you can use to test operating system, hardware and graphics performance.

For information about performance testing and benchmarks, see: Android Benchmarks

Compliance Test Suite

Google provides a suite of tests to validate that a device complies with the Android standard.

See Android Compliance Test Suite for more informaton.

Category:

- Android

Android-based Systems

About Android products, porting efforts/issues, getting root, applications, derivatives and emulators.

> From: eLinux.org

Android Hardware

Contents

Android Devices

It looks like there's a page on this on wikipedia: http://wikipedia.org/wiki/List_of_Android_devices

Hello Android has a comprehensive database as well, along with an Android app to view the database on your phone.

Mobile Phones

See http://www.androphones.com/

That have shipped

- HTC Dream
  - Branded as the G1 by T-Mobile
  - Google: Android Developer Phone 1 (ADP1) - same hardware as G1, but unlocked.
- HTC G1 V2
- HTC G2, aka HTC Magic
  - Google Android dev phone 2 - same hardware as Magic, but unlocked
- HTC Hero
  - branded as T-Mobile G2
- HTC Hero for CDMA
  - available from Sprint on Oct 11, 2009
- HTC Tatoo - Europe, Oct 2009
  - SPECS: 528MHz Qualcomm processor, 512MB ROM and 256MB RAM, 240x320 px (QVGA) 2.8-inch touchscreen, GPS, WiFi, compass and accelerometer.
  - HTC's Sense UI in TATTOO: http://www.akihabaranews.com/en/news-18854-TATTOO%2C+HTC%E2%80%99s+Latest+Android+Phone+with+Sense+UI.html
- Samsung Galaxy (in Europe)
- Samsung Behold II
  - Samsung's TouchWiz 3D Interface: http://www.akihabaranews.com/en/news-19025-Samsung+Behold+II%2C+TouchWiz+Wonder+for+T-Mobile.html
- Motorola Droid (on Verizon)
  - branded as Motorola Milestone in Europe
- Motorola Cliq
  - Has "MotoBlur" interface for interacting with social network sites
  - Wired Article Sep 2009

- Motorola Sholes ?
- Sony Ericsson XPERIA 10
  - http://gizmodo.com/5395712/sony-ericsson-xperia-x10-announced-sonys-first-android-device
- Sony Ericsson X10 Mini
  - http://www.sonyericsson.com/cws/companyandpress/aboutus/awards/award/x10miniaward?cc=gb&lc=en

Under development

- Acer Liquid
  - http://www.linuxfordevices.com/c/a/News/Acer-Liquid/
  - http://www.akihabaranews.com/en/news-19112-Acer+Finally+Announces+Their+Android+Liquid+Smartphone.html
- HTC Fiesta ?
- LG GW620
  - Engadget Article Sep 2009
  - Wired article Sep 2009
  - LG's slide out QWERTY: http://www.akihabaranews.com/en/news-18892-LG-GW620%2C+LG%E2%80%99s+First+Android+Smartphone.html

Non-phone devices

That have shipped

- OpenSourceMID.org K7 OMAP3530 MID tablet Opensourcemid
- Archos Internet tablets
  - Archos 5 and Archos 7 Internet tablets - Android tablet plays 720p video Linux For Devices Article, Sep 2009
- Android handheld game device with Cortex-A8 833Mhz powerful SoC
  - ODROID - Hackable Android handheld game device Linux For Devices Article, Dec 2009
- S3C6410 microcontroller
  - Android on S3C6410 **

Under development

- Continental AG car navigation system
- Marvel pad with targeted $100 price
- Google TV

Development board

- Devkit8500D TI DM3730 based board
- Devkit7000 Samsung S5PV210 based board
- Mini6410 Android_on_S3C6410
- MYD-AM335X TI AM335x Development Board

Category:

- Android

> From: eLinux.org

Android Porting

This page describes various effort to port Android to new boards and new processors

Contents

Porting Overview

This overview of porting steps was seen on the android-porting list: See http://www.mail-archive.com/android-porting@googlegroups.com/msg06721.html

This glosses over all the kernel work for a new board, and the android-specific kernel patches, but has some good discussion about the flash partitioning and file system bringup process.

```
If the linux kernel is up and running with all drivers in.
(particularly touchscreen and display) it shouldn't be too bad.

IHMO, the easiest way to get you running is to aggregate the initial
ramfs built into the kernel with the Android build, the root Android
root filesystem (system), and the user data section (mounted as /data
I believe) into one root filesystem.

You can then take that root filesystem as one tarball.

Modify the NAND partitioning of the kernel to set aside space for the
whole Android rootfs, and of course rebuild the kernel. (Be sure yaffs
support is in the kernel) Also no need for a ramfs at this point, just
have the kernel look to mtd2 for it's root filesystem, which will be
jffs2

Create yourself a busybox root filesystem too. Make that into a jffs2
image.

So your partitioning would look something similar to this (you'll have
to decide on the sizes of course):
mtd0: bootloader
mtd1: kernel
mtd2: rootfs (jffs2)
mtd3: Android rootfs.

Erase everything on the NAND.
Burn the normal Chumby bootloader to mtd0.
Burn the the new kernel into mtd1.
Burn the jffs2 rootfs image to mtd2.

Boot the device. Hopefully you get yourself to a prompt.

Once you have that prompt mount mtd3 to /mnt/android as a yaffs2
partition.
Untar your Android rootfs into /mnt/android.
Chown and chgrp everything under /mnt/android to "root"
chroot to that mount point "chroot /mnt/android /init"

At this point you should see Android trying to run.


I know that's a bit to chomp on, but it's more of an outline of what
you will need to do. Of course it's assuming you have the ability to
erase the whole nand and put down images amongst other assumptions,
but it should help get your mind around a little bit of the
requirements to get Android running on your device.

Regarding your bootloader question, I'd just stick with the current
one. You'll only need to modify that if/when you go into having
everything compatible with the recovery system. Which is a completely
different discussion.
```

Porting Tutorials

- Porting Android to a new Device
  - excellent and thorough paper on porting Android to the Nokia N810.
  - Has a detailed list of kernel changes and annotated diffs.
- http://wiki.kldp.org/wiki.php/AndroidPortingOnRealTarget
- Android on OMAP - excellent tutorial covering lots of different issues for porting Android to platforms based on the TI OMAP (ARM) processor
- Some cursory notes on a port to a PXA board are at: http://letsgoustc.spaces.live.com/blog/cns!89AD27DFB5E249BA!320.entry
- Adding a new device or changing the configuration of an existing device Android Device

Porting Issues

- Matt Porter (Mentor Graphics) gave a presentation on difficulties encountered while they were porting Android to MIPS and PPC processors at ELC Europe 2009. His talk was called "Mythbusters: Android" and has lots of good information.
    - See Mythbusters_Android.pdf
- Dalvik porting guide
- Matthias Brugger presented his personal "war story" on porting Android at ELC Europe 2012. See his slides on slideshare.

Android Hardware Abstraction Layer

Android talks to standard devices through its hardware abstraction layer, which overlays the kernel interfaces to devices (e.g. devices nodes, Linux system calls, etc.). To add support for your own hardware, or, in particular, to add support to Android for some new type of hardware, you need to understand this abstraction layer.

Karim Yaghmour has a good blog entry describing the Android HAL layer: http://www.opersys.com/blog/extending-android-hal

Porting to New Processors

- Mentor Graphics has ported Android to MIPS and PPC
- Power.Org supported the work to port Android to PPC
    - Nina Wilner talked about this work, and gave a demo at ELC Europe 2009
    - see Android_On_Power.pdf

Virtualization environments

There are available some virtualization environments, which allow Android applications (or the whole system) to run on other Linux-based systems, such as MeeGo or Ubuntu.

Here is some information about different systems known to exist:

- OpenMobile ACL (Application Compatibility Layer)
    - LinuxDevices article: http://www.linuxfordevices.com/c/a/News/OpenMobile-ACL-for-MeeGo/?kc=LNXDEVNL092811
    - OpenMobile product page:http://openmobile.co/products.php
- Myriad Alien Dalvik
    - LinuxDevices article: http://www.linuxfordevices.com/c/a/News/Myriad-Group-Myriad-Alien-Dalvik/
    - Myried Group page: http://www.myriadgroup.com/Device-Manufacturers/Android-solutions/~/media/D42B513FB5114FF2B4CA13A2D8CE313E.ashx
- [FIXTHIS - should add tetsuyuki presentation about running Android on Ubuntu here]

Category:

- Android

> From: wikipedia.org

Getting Root (Jailbreaking)

> Rooting is the process of allowing users of smartphones, tablets and other devices running the Android mobile operating system to attain privileged control (known as root access) over various Android's subsystems. As Android uses the Linux kernel, rooting an Android device gives similar access to administrative permissions as on Linux or any other Unix-like operating system such as FreeBSD or OS X.
>
> Rooting is often performed with the goal of overcoming limitations that carriers and hardware manufacturers put on some devices. Thus, rooting gives the ability (or permission) to alter or replace system applications and settings, run specialized apps that require administrator-level permissions, or perform other operations that are otherwise inaccessible to a normal Android user. On Android, rooting can also facilitate the complete removal and replacement of the device's operating system, usually with a more recent release of its current operating system.
>
> Root access is sometimes compared to jailbreaking devices running the Apple iOS operating system. However, these are different concepts. Jailbreaking describes the bypass of several types of Apple prohibitions for the end user: modifying the operating system (enforced by a "locked bootloader"), installing non-officially approved apps via sideloading, and granting the user elevated administration-level privileges. Only a minority of Android devices lock their bootloaders, and many vendors such as HTC, Sony, Asus and Google explicitly provide the ability to unlock devices, and even replace the operating system entirely.[1][2][3] Similarly, the ability to sideload apps is typically permissible on Android devices without root permissions. Thus, it is primarily the third aspect of iOS jailbreaking relating to giving users superuser administrative privileges that most directly correlates to Android rooting.

> From: eLinux.org

Android Hardware Fixes

Fixing Android Hardware

G1 hardware fixes

Acceleromters stopped working

If your acceleromters stop working, try the following:

- Try removing the file "akmd_set.txt" can wake it back up. Word is:
    - Remove /data/misc/akmd_set.txt and reboot your phone (or kill akmd process)

Category:

- Android

> From: eLinux.org

Android x86

Android x86 is a software port to normal **pc** s. Android x86 Homepage

The current stable release is http://www.android-x86.org/releases/releasenote-3-2-rc2 Android-x86 3.2-r2], based upon Honeycomb.

Contents

ICS 4.0.3

Build on ubuntu 11.10

see Building Android 4.0 on Ubuntu 11.10

install old version of gcc 4.4 with : sudo apt-get install gcc-4.4 g++-4.4 g++-4.4-multilib gcc-4.4-multilib

run make with : make CC=gcc-4.4 CXX=g++-4.4 -j4 iso_img TARGET_PRODUCT=generic_x86

Hardware

Fritz!Wlan

AVM GmbH AVM Fritz!WLAN N [Atheros AR9001U]

DONUT 1.6 r2

short steps to get a running system

Running the system from boot medium CD or USB-Stick

- download CD-Image android-x86-1.6-r2.iso or USB-Image android-x86-1.6-r2_usb.img.gz
- burn cd image or for the usb-image use the following commands on a linux box "gunzip http://www.android-x86.org/download" and "dd if=android-x86-1.6-r2_usb.img of=/dev/sda" (of=/dev/sda is depending on where your linux mounted your usb-stick)
- boot from created medium and choose the first menu entry "Live USB - Run Android-x86 without Installation"
- in android goto settings/sound & display/screen timeout and set to "never timeout"

run the system from harddisk

s. as well the installation section on android-x86.org

- boot from CD or USB-Stick android x86 boot menu choose the fourth option "Installation - Install Android-x86 1.6-r2 to harddisk"
- select "Create/Modify partitions" and create a bootable partition with cfdisk
- select created partition e.g. sda1 and format with e.g. ext3
- install GRUB by selecting 'yes'
- reboot system and boot from harddisk and select the default menu entry "Android-x86 1.6-r2"
- in android goto settings/sound & display/screen timeout and set to "never timeout"

details for a running system

SD-card

for mounting an sd card see [1]

navigation

- HOME \<- windows key left
- BACK \<- esc
- MENU \<- menu-key

touchscreen or mouse

touch the right end of statusbar to activate or deactivate the following functions

- HOME \<- touch status bar
- MENU \<- touch statusbar from left to right.
- BACK \<- touch t statusbar from right to left.

sound

Keys on ASUS EeePC

```
Fn-F7, F8, F9,
some models are
Fn-F10, F11, F12
```

Notebooks

```
Some notebooks also have volume adjustment hotkeys
```

raise sound volume in shell

```
* change screen to console 1, press Alt+F1
* enter
  "alsa_amixer cset numid=1 31" for 'Front Playback Volume' and/or
  "alsa_amixer cset numid=20 31" for 'Master Playback Volume'  and/or
  "alsa_amixer cset numid=3 31" for 'Speaker Playback Volume'
* go back to graphic screen,  press Alt+F7
```

x86 PCs with normal keyboard

rear panel audio jack and front panel audio jack depend on the setting of 'Front Playback Volume' alsa_amixer sound setting

video

application errors

- menu /settings/about phone/System tutorial -> Sorry! The application settings (process com.android.settings) has stopped unexpectedly. Please try again.

Development

improvements/request for changes

- set "settings/sound & display/screen timeout" to "never timeout" as for an x86 system there is no need to timeout an new users don't know what happens.

errors

shell

navigation/keyboard

keyboard layouts see /system/usr/keylayout/

command to get events: getevent

Keymaps and Keyboard Input, a detailed description

File Infos

1. system.sfs - squash filesystem
2. system.img - ext2 file Image
3. ramdisk.img - gzip cpio file - extract in an empty folder with "gzip -d \< ramdisk.img |cpio -id"
4. initrd.img - gzip cpio file - extract in an empty folder with "gzip -d \< initrd.img |cpio -id"

hand made changes on an usb-Stick Android 1.6 r2

File content of an usb-stick

```
├── android-system
|   ├── initrd.img
|   ├── install.img
|   ├── kernel
|   ├── ramdisk.img
|   └── system.sfs
├── android-x86.xpm.gz
├── cmdline
├── grub4dos
├── kernel -> grub4dos
├── lost+found
├── menu.lst
└── ramdisk
.
2 directories, 11 files
```

steps to change files in system.sfs (system.img)

```
* Ubuntu 10.4 box
* change to shell, press strg+alt+F1
* sudo -i
* aptitude and install squashfs-tools
* modprobe squashfs
* cd /home/administrator
* copy system.sfs (squash file system) to harddisk, in /home/administrator
* mkdir systemsfs
* mount ./system.sfs ./systemsfs -t squashfs -o loop
* copy ./systemsfs/system.img /home/administrator/
* mkdir systemimg
* mount ./system.img ./systemimg -t ext2 -o loop
```

now cd to systemimg directory and make the changes

eg. change *.kl files for sound F7 (scanncode=65) = mute; F8 (scanncode=66) = volume_down; F9 (scanncode=67) = volume_up

```
* cd usr/keylayout
* vi *.kl
* change lines
  key 113   VOLUME_MUTE
  key 114   VOLUME_DOWN
  key 115   VOLUME_UP

  key 65   VOLUME_MUTE
  key 66   VOLUME_DOWN
  key 67   VOLUME_UP
```

sound

s.

1. http://wiki.archlinux.org/index.php/Advanced_Linux_Sound_Architecture
2. http://www.alsa-project.org/main/index.php/Main_Page

yaffs2 - filesystem

```
* download unyaffs2 http://code.google.com/p/yaffs2utils/downloads/list s. description http://code.google.com/p/yaffs
2utils/ ; extract to yaffs2util
* download snapshot as described in http://yaffs.net/node/346, extract the source file to directory yaffs2
* change ~/yaffs2util/Makefile with vi and set "KERNELDIR   = /usr/src/linux-headers-2.6.32-21" ; depending on the lo
cation of your header files
* execute make
* goto subfolder ~/yaffs2util/src and copy mkyaffs2 and unyaffs2 to /home/administrator/bin
* execute: PATH=$PATH:/home/administrator/bin
*
```

Links

http://androidoniphone.blogspot.com/2010/04/install-android-on-iphone-guide.html http://android-dls.com/wiki/index.php?title=Main_Page

Category:

- Android

> From: eLinux.org

Android Applications

This page has some miscellaneous applications and services which may be of interest to developers:

DLNA

A fast growing open-source stack for Android is

- Cling - http://www.teleal.org/projects/cling/

Alternatively you might want to look at

- http://www.cybergarage.org/twiki/bin/view/Main/CyberLinkForJava

Category:

- Android

> From: eLinux.org

Android Derivatives

This page has information about systems that are derived from Android.

Why non-Google Android

- http://www.reliableembeddedsystems.com/pdfs/2012-07-03-penguin-and-droid.pdf
    - presentation by Robert Berger discussing non-Google Android markets

Cyborgstack

This project aims to utilizes the Android system in other product categories, including deeply embedded:

- web site: http://www.cyborgstack.org
- wiki: http://wiki.cyborgstack.org
- github: http://github.com/cyborgstack
- twitter: @cyborgstack
- list: http://lists.cyborgstack.org/listinfo/dev

Headless Android

From: https://lkml.org/lkml/2012/2/15/443

As an FYI, I thought some of you might be interested in knowing that Android's user-space can be modified to run on headless systems (i.e. without a framebuffer.) IOW, you can configure the FB stuff completely out or have a kernel port that doesn't have an FB (yet?) and still run the Android use-space.

I've put this up as part of the Cyborgstack project:

```
$ repo init -u git://github.com/cyborgstack/android.git -b headless
$ repo sync
$ ...
```

The relevant presentation from the Android Builders Summit is here: cyborgstack-120213.pdf

Essentially, Headless Android is the AOSP but WITHOUT:

- SurfaceFlinger
- WindowManager
- WallpaperService
- InputMethodManager

It gives you is all the Android framework but for ui-less systems (no FB.) What it means, is that, save for Activities, you can use the standard Android development tools (Eclipse, SDK/NDK, etc.) to create apps that use:

- ContentProviders
- Services
- BroadcastReceivers

Why would you want this instead of using "Embedded Linux"? Honestly I was very skeptical when some developers first mentioned to me that they were interested in doing this. I was in fact very dismissive of it. But I kept getting more and more inquiries about this. So I decided to bite the bullet and give it a try.

Now that I have, I think there are 2 clear benefits to using this instead of "embedded Linux": 1) you get one platform for all your device development, whether it has a UI or not 2) your devices become programmable by any developer that knows the Android API (and, as you may know, there's growing number of those.)

That said, what I've done is very much a proof of concept. It's in fact a dirty hack at this point. Please don't ship this just yet. It needs a lot more eyeballs and certainly a lot more work. But, it's good enough to give you a taste of what's possible and allow you play with it.

Cheers,

-- Karim Yaghmour

Category:

- Android

From: eLinux.org

Linux emulators for Android

See also

- KBOX Linux emulator for Android

Android Community

The news, events, mailing list, people and organizations for Android Developers.

> From: eLinux.org

Android News

This page has links to news about the Android platform

Contents

Android News Portals

- http://androidguys.com/ - General news
- http://androinica.com/ - Android Blog

Interviews

- Google's Rubin: Android 'a revolution' - CNet News May 2009

"Android, Google's mobile operating system, doesn't generate revenue for the company, and likely never will--at least in the direct sense. But Andy Rubin, Google's director of mobile platforms, thinks Google and the world will benefit from any device created with the intent of getting more people onto the Internet, and isn't shy about explaining why the open-source approach chosen for Android holds the most promise of reaching that goal."

Industry/Consortium news

Mentor Graphics acquires Embedded Alley

Mentor Graphics, a large EDA firm, acquires the embedded Linux consulting company Embedded Alley, in July, 2009.

- See http://www.linuxfordevices.com/c/a/News/Mentor-Graphics-acquires-Embedded-Alley/

OESF

The Open Embedded Software Foundation (OESF) is an Android Consortium formed in Japan in February, 2009. It included 23 companies at startup, and has the support of Google.

- See http://techon.nikkeibp.co.jp/english/NEWS_EN/20090325/167661/

Software

Releases

- Android SDK version 3.0 (Honeycomb)
- Android SDK version 2.3.3
- Android SDK version 2.3 (Gingerbread)
- Android SDK version 2.2 (Froyo)

- Android SDK version 2.1 (Eclair 2.1) released (Jan 12, 2010)
  - http://www.techtree.com/India/News/Android_21_SDK_Released/551-108663-580.html
- Android version 1.6 (Donut) released (version 4 released December 2009)
  - http://developer.android.com/index.html
  - http://android-developers.blogspot.com/search/label/Android%201.6
  - http://developer.android.com/sdk/android-1.6-highlights.html
- Android version 1.5 (Cupcake) released (April 2009)
  - Android 1.5 'Cupcake' starts arriving on G1s ComputerWorld, April 30, 2009
  - Android 1.5 firmware available LWN.net, April 27, 2009
    - HTC (the manufacturer of the Android Dev Phone) has posted a set of Android 1.5 images for the ADP1. There's also reasonably straightforward instructions on flashing those images into a phone

Ports

- Android ported to MIPS Linuxdevices, April 2009
  - Embedded Alley has ported Android to the Alchemy MIPS processor.
- Android ported to various ARM processors (ARM9, ARM11 and Cortex-A8 based). Including e.g. OSK board, Nokia N800, Nokia N810, BeagleBoard, Devkit8000 and Gumstix Overo board.
- Devkit7000 Samsung S5PV210 based board.
- Devkit8500D TI DM3730 based board.

Videos

- WLAN and Bluetooth demo on DM3730 running Android featured on You Tube
- Android demo on DM3730 featured on You Tube
- Android Demo on Mistral's TMDSEVM3530 featured on You Tube
- Android Demo on Mistral's TMDXEVM3503 featured on You Tube

Features

Interfaces

- SenseUI (HTC)
- MotoBlur (Motorola)
- TouchWiz (Samsung)
- Rachael (Sony Ericsson)

Applications

- Google Maps (with turn-by-turn navigation)
- Google Goggles - search by image from camera
  - http://www.pcmag.com/article2/0,2817,2356786,00.asp
    - Hands off with Google Goggles

- DoCoMo Augmented Reality Concept running on Android : http://www.akihabaranews.com/en/news-18527-DoCoMo+Augmented+Reality+Concept+at+Wireless+Japan+09.html

Phone news

- Samsung phone
  - Samsung's Android phone to launch in Germany Linuxdevices, April 2009
- T-mobile ships more than 1 million G1 phones
  - T-Mobile sells a million Android phones Linuxdevices, April 2009

Netbooks and MIDs

- ARM-based MIDs run Android, report says Linuxdevices, April 27, 2009
- First Android netbooks surface Linuxdevices, April 2009
  - ARM-based netbook by Skytone announced.

Category:

- Android

> From: eLinux.org

Android Events

The only events I know of (as of Jan 2010) with Android content are:

Contents

Google Events

- Google I/O - see the Google I/O wikipedia entry
  - Google I/O for 2011 was in on May 10,11 in San Francisco
    - See Tims GoogleIO 2011 Notes
- Google Developer Days
  - Regional events with Google developers

Android Builders Summit

This is a new event (in 2011) for Android systems developers. This includes developers who are porting Android to new hardware, working on the Android kernel or low-level infrastructure, or otherwise working on Android itself (not on Android application development).

- ABS home page
  - Presentation slides

Other Non-Google Events

- Android Open Conference
  - October 9-11, San Francisco
  - This is an O'Reilly event, and appears to be trying to cover the whole Android ecosystem.
- AnDevCon II - November 6-9, San Francisco
  - Covers mostly application development, but has other sessions as well.

Linux events

- http://events.linuxfoundation.org/events/embedded-linux-conference/ Embedded Linux Conference
  - See http://embeddedlinuxconference.com for an overview of multiple years
    - There were android talks especially at ELC Europe 2009 and 2010
    - See Matt Porter's talk and Nina Wilner's talk at ELC Europe 2009 Presentations

        - See Tim Bird's talk and the Android tutorial at ELC Europe 2010 Presentations
- Some CELF Japan Jamborees have technical content related to Android
    - The following had Android presentations: Japan Technical Jamboree 31, Japan Technical Jamboree 34

Regional events

Upcoming...?

- Droidcon London 2010 - October 28,29, 2010, London
- Droidcon Belgium - next one = January 21, 2011, Brussels
- [http://www.android-group.jp/abc2011w/ Android Bazaar and Conference - Jan 9, 2011, Tokyo)

Related Conferences

- Game Developer's conference is starting to see Android participation
    - See http://android-developers.blogspot.com/2010/01/android-at-2010-game-developers.html

Previous Events

- Android Developer's Conference - March 7-9, 2011
    - This seems to be the first full-blown Android developer event, covering mostly application development, but also some system talks
    - archive online at???

Regional events

- [http://www.android-group.jp/abc2011w/ Android Bazaar and Conference - Jan 9, 2011, Tokyo)
- Droidcon Belgium - next one = January 21, 2011, Brussels
- Droidcon London 2010 - October 28,29, 2010, London

Categories:

- Android
- Events

> From: eLinux.org

Android Web Resources

Contents

Web Sites

Google sites

- http://developer.android.com/ - site mainly for application developers
  - mailing lists for app developers
- http://source.android.com/ - site devoted to source code management
  - mailing lists for source/platform discussions

Community sites

- Android-DLs
  - Android-DLs wiki
  - Android-DLs forum
- xda-developers
  - XDA Developers Dream Android Development forum - lots of good discussion about ROMS and Android system development
- AndroidCommunity.com
- Stack Overflow (android-tagged questions)
- Android Freeware Lovers - community driven repo of open source and freeware apps
- Android Freeware Android freeware or on sales apps

Other sites

- Android™ on OMAP
- Android Demo on Mistral's TMDSEVM3530 featured on You Tube
- Android Demo on Mistral's TMDXEVM3503 featured on You Tube

Mailing Lists

See http://developer.android.com/community/index.html for a list of official Android mailing lists, and instructions for joining these lists.

- android-platform
  - Discussion of user-space Android code, including system libraries, service, public APIs or built-in applications.
- android-porting
  - Toolchains, device-specific customizations, board bring-up and optimizations
- android-kernel

  - Discussions of Linux kernel and Android-related changes
- repo-discuss - Repo and Gerrit discussion
- android-beginners
  - Beginning users of SDK and getting started with Android development
- android-developers
  - Experienced developers - implementation help, optimizations, etc.
- android-discuss
  - General discussion of the platform, usage, ideas for new features, etc.
- android-security-discuss
  - Security issues. Please don't disclose vulnerabilities directly on this list.
- android-security-announce
  - Low-volume list for security announcements by the Android Security Team.
- Android Market Help Forum
  - A web-based discussion forum for Android Market questions.

Other lists

- android-ndk - Android Native Development Kit (NDK) mailing list
  - List for people working on native code. This list is targeted by Google at supporting developers who are writing native shared libraries to use via JNI from their Java-based applications. You won't get any help using 'agcc' or other community-developed native development tools there.

Category:

- Android

> From: eLinux.org

Android People

Here are some of the key Android developers you may encounter on mailing lists or at events:

Google People

See Google's listing of the Android core technical team

- Dianne Hackborn - platform programmer, worked on original OpenBinder
- Brian Swetland - lead kernel developer
  - See http://lkml.org/lkml/2009/6/10/397
- Arve Hjønnevåg - kernel developer (original submitter of wakelock patches)
- Mike Chan - Droid developer
- Dan Bornstein - Dalvik creator
- Chris DiBona - Open Source Programs Manager at Google

Non-google People

- Karim Yaghmour - President of Opersys, long-time Embedded Linux and Android expert, author and trainer
- Greg Kroah-Hartman - long-time kernel developer, sometimes responds on android-kernel mailing list
- Tim Riker linked-in account - CTO at Saygus, maker of an Android-based video phone
- Tetsuyuki Kobayashi - Developer at Kyoto MicroComputer. Has an excellent blog with several great articles about Android. http://kobablog.wordpress.com/
  - presentation: Android is not just Java on Linux
    - an excellent overview of Android
  - page of presentations by Kobayashi-san: http://www.slideshare.net/tetsu.koba/presentations

Hacker community

- Steve Kondik (also known as Cyanogen) - creator and maintainer of the cyanogenmod custom images
- Benno - has a blog, provides busybox binary (V.P. of Engineering at Open Kernel Labs)
  - Android Framework Startup tutorial on framework startup (and how to change a ramdisk.img)
  - Android - under the hood analysis of system startup
  - list of android articles by Benno
  - busybox binary
- JesusFreke - Used to make custom system images. Author of smali/baksmali (dex assembler/disassembler)
  - JesusFreke's Blog (with release anouncements)
  - smali project page
- Haykuro - makes custom system images
  - Haykuro's site (blog, info)
- Brian Gupta (aka Brandorr) - author of FAQ at Android-DLs
- TheDudeOfLife - makes custom system images
  - announcement of 1.5 (cupcake) build
- Others on xda-developers: xmoo, Stericson, dapro, marcusmaximus, manup456

Category:

- Android

> From: eLinux.org

Android Organizations

Here are some Android organizations

- Open Handset Alliance - the original group of companies organized by Google to collaborate on Android technologies and delivery

Regional

- Korean Android Platform Group
    - Have published (or are endorsing) a book: Inside Android
- Android Association of Japan (or Japan Android Group)
- http://www.yokohama.android-pf.org/ Yokohama Android Platform Club

Category:

- Android

Hardware Pages

Resources or information about the popular embedded boards.

From: eLinux.org

Beagleboard:Main Page

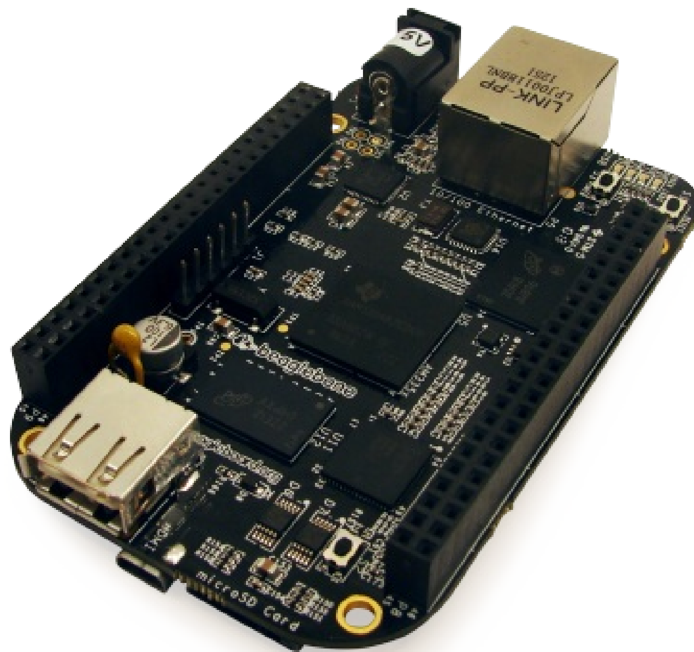
(Redirected from BeagleBoard)

Contents

Welcome to the BeagleBoard.org Support Page

This is the place where BeagleBoard fans can come to find information on the BeagleBoard,BeagleBoard-xM, BeagleBone and BeagleBoneBlack projects that are manufactured by CircuitCo. Most of this information can also be found on the beagleboard.org site and is provided here as a convenience. When new revisions of hardware or software is released, you may be able to find information here sooner than it appears on the beagleboard.org site. You may find information here not found on the beagleboard.org site as well.

Click on any of the following boards to go to its official wiki page



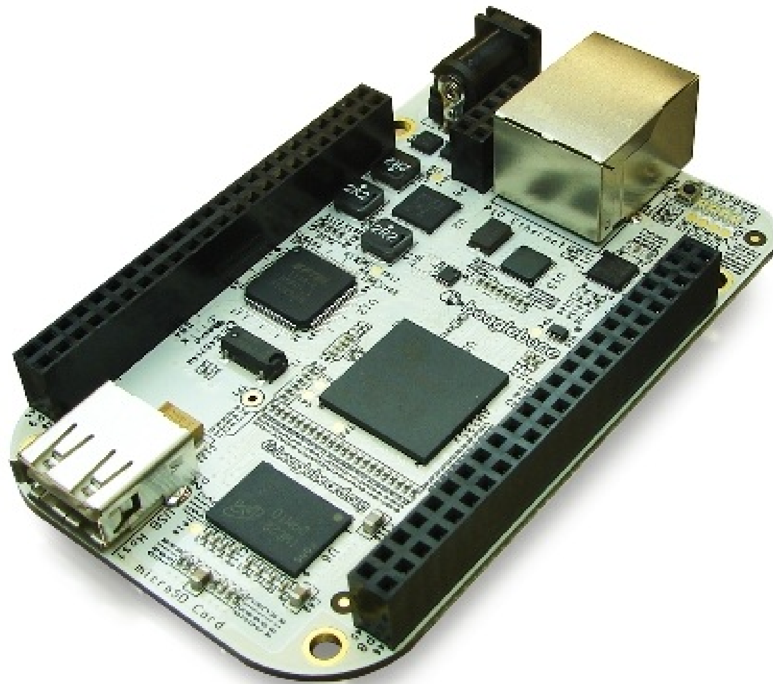
BeagleBone Black NEW

The latest addition to the BeagleBoard line. Similar to the BeagleBone but with 1GHZ AM335X processor, 512MB of DDR3, 2GB eMMC, and HDMI.

 Go To Official Wiki Page - Go here to find official specifications and resources

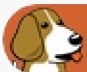
 Go To Community Wiki Page - Go here to find community created "howto" pages and resources

 Go To Capes Wiki Page - Go here to find a registry of compatible acessory board Capes



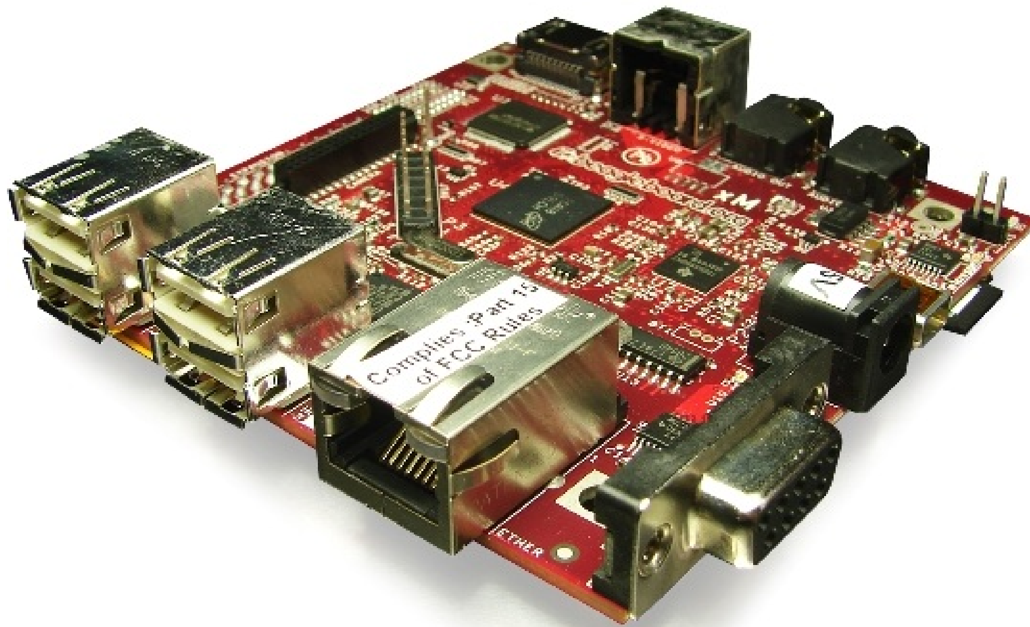
BeagleBone

A bare-bones BeagleBoard that acts as a USB or Ethernet connected expansion companion for your current BeagleBoard and BeagleBoard-xM or works stand-alone.

 Go To Official Wiki Page - Go here to find official specifications and resources

 Go To Community Wiki Page - Go here to find community created "howto" pages and resources

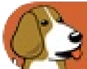
 Go To Capes Wiki Page - Go here to find a registry of compatible acessory board Capes



BeagleBoard-xM

Delivers extra ARM ® Cortex TM -A8 MHz now at 1 GHz and extra memory with 512MB of low-power DDR RAM, enabling hobbyists, innovators and engineers to go beyond their current imagination.

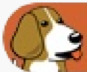
 Go To Official Wiki Page - Go here to find official specifications and resources

 Go To Community Wiki Page - Go here to find community created "howto" pages and resources



BeagleBoard

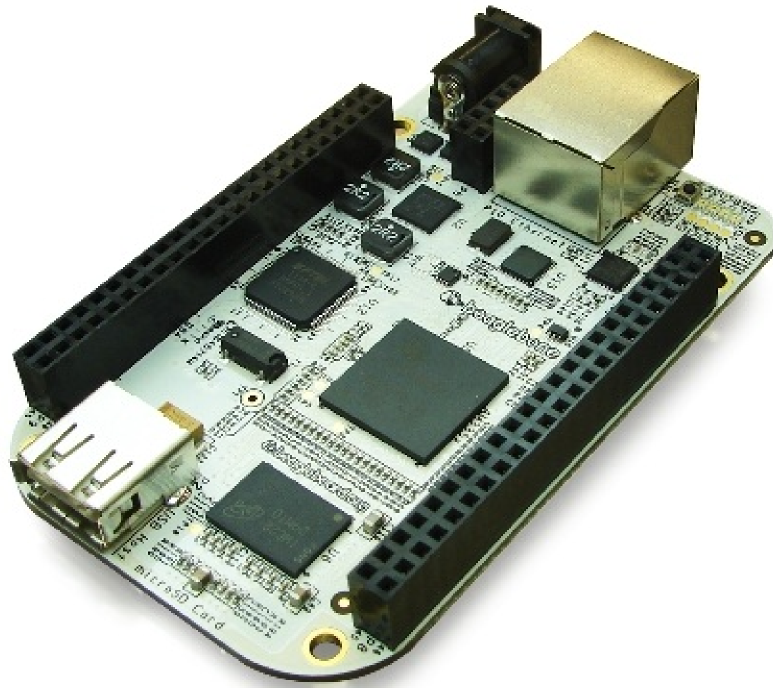
A low-cost, fan-less single board computer that unleashes laptop-like performance and expandability without the bulk, expense, or noise of typical desktop machines.

 Go To Official Wiki Page - Go here to find official specifications and resources

 Go To Community Wiki Page - Go here to find community created "howto" pages and resources

> From: eLinux.org

BeagleBone



BeagleBone

A bare-bones BeagleBoard that acts as a USB or Ethernet connected expansion companion for your current BeagleBoard and BeagleBoard-xM or works stand-alone.

 Go To Official Wiki Page - Go here to find official specifications and resources

 Go To Community Wiki Page - Go here to find community created "howto" pages and resources

 Go To Capes Wiki Page - Go here to find a registry of compatible acessory board Capes

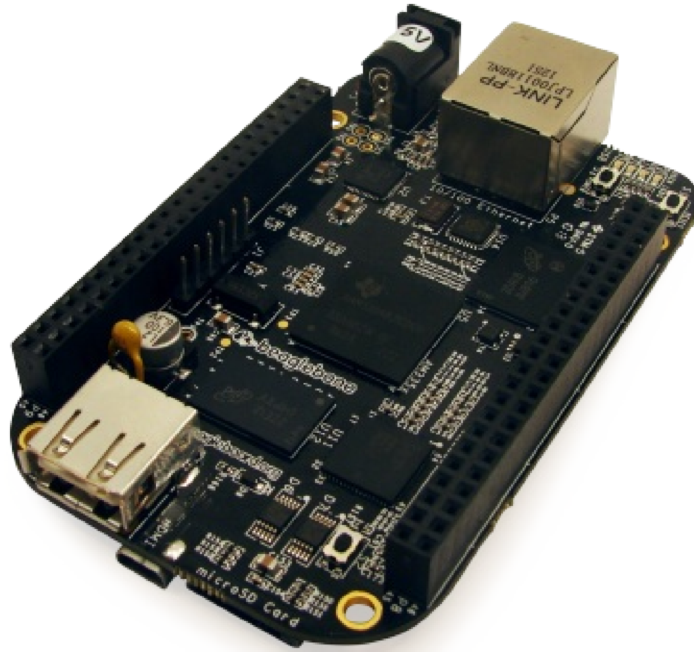
See also: BeagleBoneBlack

Category:

- BeagleBone

> From: eLinux.org

BeagleBoneBlack



BeagleBone Black NEW

The latest addition to the BeagleBoard line. Similar to the BeagleBone but with 1GHZ AM335X processor, 512MB of DDR3, 2GB eMMC, and HDMI.

 Go To Official Wiki Page - Go here to find official specifications and resources

 Go To Community Wiki Page - Go here to find community created "howto" pages and resources

 Go To Capes Wiki Page - Go here to find a registry of compatible acessory board Capes

> From: eLinux.org

Beagleboard:BeagleBone Capes

BeagleBone is a credit-card sized expandable Linux computer that connects with the Internet and runs software such as Android 4.0 and Ubuntu. It allows developers to evaluate the Sitara™ AM335x ARM® Cortex™-A8 processors with a single cable and 10-second Linux boot-enabling development in less than five minutes. Adding cape plug-in boards to the popular BeagleBone computer allows hobbyists, makers and developers to quickly and easily augment BeagleBone's capabilities with LCD screens, motor control and battery power as well as the ability to create their own circuits.

Contents

Click here to view the Production video for BeagleBone Black

Compatibility Between Different Capes Matrix

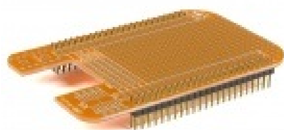
To view the compatibility between capes that are listed in this catalog, please download the compatibility matrix below.

- File:BeagleBone-Cape-Compatibility-Matrix-12-09-11.xls

BeagleBone Black Compatibilty Matrix

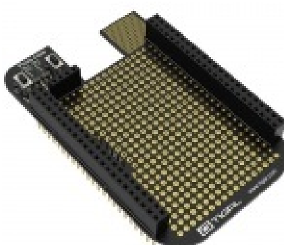
for information on BeagleBone Black Compatibility visit the Beagleboard:BeagleBone Black Capes wiki page

BeagleBone Capes Catalog



BeagleBone Proto Cape Logic Supply





BeagleBone ProtoCape

Tigal





BeagleBone eeProtoCape Logic Supply





BeagleBone Breadboard

BeagleBoardToys





Proto Cape Kit Adafruit



Basic Proto Cape BeagleBoardToys





Generic Proto Cape BeagleBoardToys

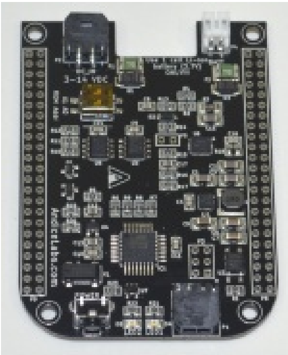




Extended Proto Cape

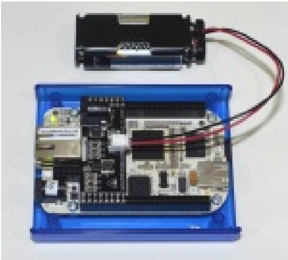
BeagleBoardToys





PowerCape AndiceLabs





PowerBar AndiceLabs





Power Supply Cape BeagleBoardToys





BeagleBone Battery Cape BeagleBoardToys

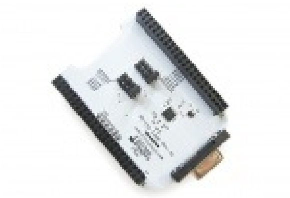




BeagleBone TTL-RS232 Micro Cape

Logic Supply





RS232 Cape





Audio Cape

BeagleBoardToys





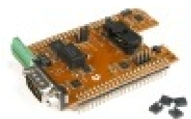
BeagleBone Weather Cape BeagleBoardToys (NEW REVISION)





TT3201 CAN Cape TowerTech





BeagleBone Serial CAN RS485 RS232 Cape

Logic Supply





CAN Bus Cape BeagleBoardToys

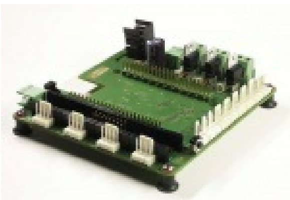




RS485 Cape

BeagleBoardToys





BeBoPr Cape AES electronics



BeBoPr-Plus AES Electronics





BeBoPr++ www.bebopr.info





Osso Cape Nexlab





Bacon Cape BeagleBoardToys





BeagleBone Ninja BeagleBoardToys



BeagleBone ROV BeagleBoardToys



Inertial Navigation System Cape

UAV Navigation





BeagleIO Impressx





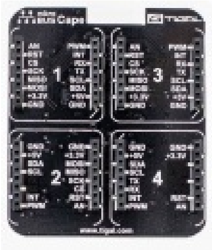
USB HUB Cape Terratechnos,Inc.



Arduino Shield Adapter

Impressx





BeagleBone mikroBUS Cape

Tigal





Smart Dual Relay Cape

Sierra Foxtrot





BeagleBone M2M Cape Yantrr Electronic Systems





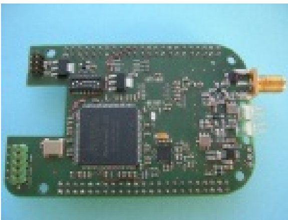
Relay Cape BeagleBoardToys





BeagleBone Dual Relay Plus IO Logic Supply





Radarcape Guenter Koellner



ADC Cape BeagleBoardToys





Escherlogic DogHouse

Escherlogic





BeagleBone 8 Port 1-Wire

BY Innovation, Inc.



Nimbelink Skywire Cellular Modem Cape Nimbelink Corp





BeagleBone GPS/GPRS EACH-CS





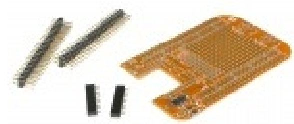
INTERNET HUB Cape EACH-CS





Tracking Cape Ciudad Oscura





BeagleBone XBee proto cape Logic Supply





BeagleBone XBee Cape Dr. Phil Polstra





RTC Cape BeagleBoardToys





PWM Cape BeagleBoardToys





MotoCape BeagleBoardToys





MiniDisplay Cape BeagleBoardToys





BeagleBone LCD3 Cape BeagleBoardToys





BeagleBone LCD4 Cape BeagleBoardToys





BeagleBone LCD7 Cape BeagleBoardToys





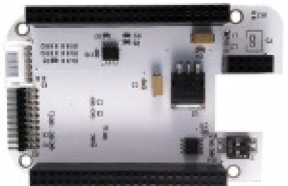
4D 4.3 LCD CAPE 4D SYSTEMS





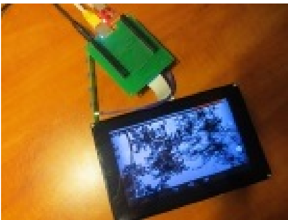
Nh7Cape Cembsoft





BeagleBone LVDS LCD Cape

Chalkboard Electronics



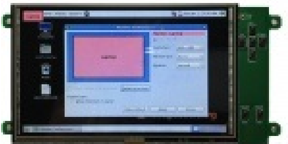
Alcdcape Cembsoft





nh5cape Cembsoft





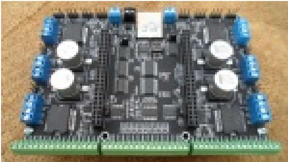
WTLCD050R LCD cape WITZTRONICS





BeagleBone 4Gb 16-Bit NAND Module

BeagleBoardToys



BeagleBone Robo Cape EmbeddedToys





BeagleBone Motor w/ NXT Connectors

BeagleBoardToys



BeagleBone Motor w/ Screw Blocks

BeagleBoardToys



BeagleBone MSTP Cape Plano CAD



BeagleBone DVI-D Cape BeagleBoardToys



BeagleBone VGA Cape BeagleBoardToys



BeagleBone RF Cape BeagleBoardToys



BeagleBone Memory Expansion Cape

BeagleBoardToys



BeagleBone TiWi-5E w/ Chip Antenna

BeagleBoardToys



BeagleBone TiWi-5E w/ EXT. Antenna

BeagleBoardToys



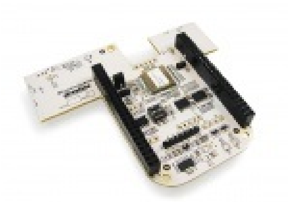
BeagleBone TiWi-BLE w/ Chip Antenna

BeagleBoardToys



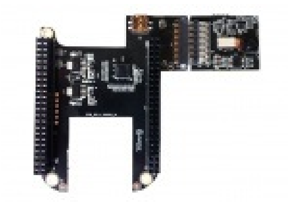
BeagleBone TiWi-BLE w/ EXT. Antenna

BeagleBoardToys



WL1835MOD w/ Chip Antenna

BeagleBoardToys



BeagleBone HD Camera Cape

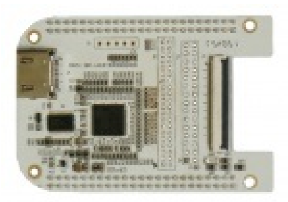
RadiumBoards





BeagleBone 3.1MP Camera Cape

BeagleBoardToys



BeagleBone-HDMI CAPE

Embest



BeagleBone DVI-D w/ Audio Cape

BeagleBoardToys



BeagleBone DVI-D Cape BeagleBoardToys



Industrial I/O Cape iCOM Qinno



BBB-GVS land-boards





CryptoCape SparkFun Electronics





BeagleBone 128Mb 16-Bit NOR Module

BeagleBoardToys



BeagleBone ProfiBus Cape

BeagleBoardToys

Capes Listing

If you have a cape that will be in production or able to be purchased and you would like it listed here, please create a wiki page for your cape on eLinux.org and contact support@circuitco.com with the link. We will add your cape to this registry page. You can use http://elinux.org/CircuitCo:BeagleBone_LCD3 as a template (to view the wikitext go to the link and click on the "View Source" or "Edit" tab on the top right corner). Also, submit any patches to the kernel, including .dts files for your cape, to the BeagleBoard group/mailing list.

Note: if you don't have a eLinux.org account, you will need to create one in order to create or edit a wiki page. Note: if you have any question related to a particular cape, please contact the designer of that cape. For other questions regarding this page, please contact us at support@circuitco.com

To go back to BeagleBone wiki, please click here.

To go back to BeagleBone Black wiki, please click here.

> From: eLinux.org

Minnowboard:MinnowBoard

(Redirected from MinnowBoard)



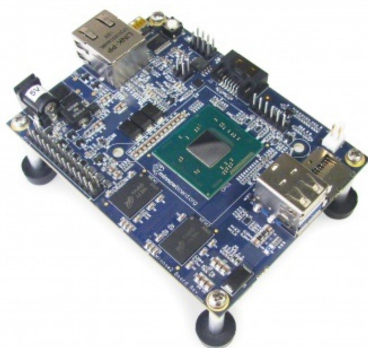
NEWS: Interested in Google Summer of Code with MinnowBoard.org? Please visit our MinnowBoard GSoC2015 Page!

Introduction

http://www.minnowboard.org is a community dedicated to the support of Open Hardware utilizing Intel® processors. This web site will serve as a starting point for the MinnowBoard community.

The MinnowBoard is an Intel® Atom™ processor based board which introduces Intel® Architecture to the small and low cost embedded market for the developer and maker community. It has exceptional performance, flexibility, openness and standards

Boards

MinnowBoard MAX	MinnowBoard
	

Community

- IRC: Freenode network, Channel: #minnowboard
- Mailing List archives

Categories:

- CircuitCo
- MinnowBoard

> From: eLinux.org

RPi Hub

Notice: The Raspberry Pi Wiki pages on this site is collaborative work - the Raspberry Pi Foundation is **not** responsible for content on these pages.

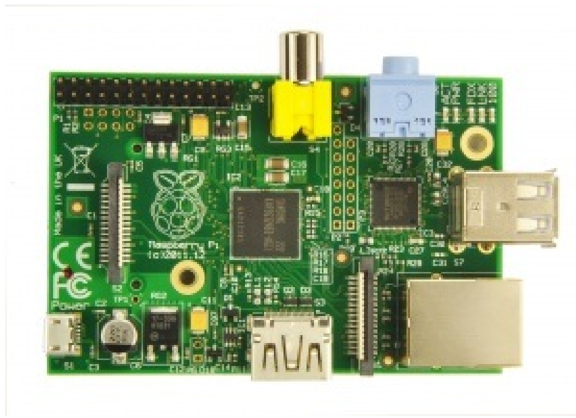
Contents

	Raspberry Pi Wiki Hub deu | eng | fra | pt-br

Now shipping to customers

See the Buying Guide on how to order one, or visit the Raspberry Pi Foundation Home Page

About



The Raspberry Pi production board (model B Rev 2.0)

The Raspberry Pi (short: RPi or RasPi) is an ultra-low-cost ($20-$35) credit-card sized Linux computer which was conceived with the primary goal of teaching computer programming to children. It was developed by the Raspberry Pi Foundation, which is a UK registered charity (Registration Number 1129409). The foundation exists to promote the study of computer science and related topics, especially at school level, and to put the fun back into learning computing. The device is expected to have many other applications both in the developed and the developing world (Read more).

Raspberry Pi is manufactured and sold in partnership with the worldwide industrial distributors Premier Farnell/Element 14 and RS Components, and the Chinese distributor Egoman Technology Corp[1].

- You can get the latest news from the Foundation Home Page, the Twitter Feed or in the forums.
- For Raspberry Pi frequently asked questions see the FAQ section or the Raspberry Pi Foundation's FAQ page.
- Both manufacturing partners provide community areas for more technically focused discussions, articles, FAQs and related information:

- Premier Farnell: Element 14 Raspberry Pi Group

- RS-Components: DesignSpark - Raspberry Pi

- Products are RoHS, CE, FCC, CTick, CSA and WEEE compliant[2]. In common with all Electronic and Electrical products the Raspberry Pi should not be disposed of in household waste. Please contact the distributor from whom you purchased your Raspberry Pi device for details regarding WEEE in your country.

- Price: 20USD Model A+, 35USD for Model B+, excluding taxes, postage and packaging. For information about availability and shipping see the Buying Guide.

History

If you are interested in why the Raspberry Pi was created, and why it is what it is, check the General History page, which highlights relevant events in its history. It is not intended to be a detailed history, so it can be read quickly. You could also check the design changes page for how the Raspberry Pi has evolved, and the manufacturing differences page that may help if you are having problems with your board.

Getting Started

Buying Guide

Where can I get one and for how much?

- Base price is $20 for the model A+ and $35 for the Raspberry Pi 2. This price excludes local taxes and shipping.
- The Raspberry Pi's official worldwide distribution partners are Premier Farnell/Element 14 and RS Components
- Detailed information and other resellers can be found on the Buying Guide page.
- You can find out which peripherals and such are tested to work with the Pi in the Verified Peripherals section

Basic Setup

First little Raspberry Pi Steps...

- Ensure you have all the equipment you need to go with your Raspberry Pi.
- Become familiar with the board layout and connect it ready for power up.
- If you have not been provided with a pre-setup SD card you will need to prepare one with your chosen Operating System distribution
- If you are not using a HDMI monitor you may need to set up the correct video mode by editing the RPiconfig text file on the SD-card.
- Note: On the Debian OS after you log in you need to type **startx** at the prompt to get a graphic desktop.
- Particularly after first boot its important to do a clean shutdown with the command **sudo halt**
- **Having problems? Try the Troubleshooting page.**

Beginners Guide

You've just got your new Raspberry Pi device - what now?

- Beginners Guide
- Learn about the basics with the H2G2 - Introducing the Raspberry Pi entry.
- Read a small book for the Raspberry Pi Beginner [1]
- Get started with some basic projects and tutorials:
  - Raspberry Pi YouTube Tutorials
  - Raspberry Pi IV Beginners
  - My First Raspberry Pi Game
  - Guides, tutorials, tools and distribution downloads
- Easy GPIO Hardware & Software - in-progress at the moment
- Take a look through the Community section, which contains a range of beginner and advanced tutorials and guides, as well as groups to help you find like-minded developers.
- Pick up a copy of the Raspberry Pi Handbook to get you started on some fantastic projects
- Get started with Linux: Linux Basics

Resources

Hardware & Peripherals

- The Model B is more advanced than the Model A -

Software & OS Distributions

The Raspberry Pi will run a range of OS Distributions and run a variety of software.

- See Software for an overview, and OS Distributions for supported operating

Documentation

Datasheets

IC Datasheets and schematics links page.

Datasheets organised by category from the Frambozenier.org projec

see RPi Hardware.

- The RPi can be plugged into a suitable TV or monitor.
- The unit will support a range of devices, peripherals and accessories.
- The Low-level interfaces allow the use of optional Expansion Boards in a wide range of projects.
- The Foundation has launched a camera module with a 5MPixel sensor capable of capturing video at 30fps at 1080p
- Serial port connection instructions
- GPIO interface circuits for connecting switches, relays, etc
- For more advanced issues including see Advanced Setup.
- Setting up peripherals - examples/HowTos
- List of boards and user feedback
- Power Supply construction - HowTo
- Comparison to other hardware

system and pre-configured 'images'.

- Officially supported OS distributions include Raspbian, Arch Linux and RISC OS Open.
- Many unofficial distributions are available on the Distributions page.
- Advice is also available if you want to compile a kernel, boot from the network using U-Boot, or test the Pi's performance.
- The Raspberry Pi supports a wide range of programming languages, with many tutorials available.
- Information about installing. specific applications is available through the link.
- Extensive (boot) configuration info (config.txt) is available here.
- Information about various utilities that can be used with your Raspberry Pi can be found here.

Datasheets organised by category from the Frambozenier.org projec

Troubleshooting

Head over to the troubleshooting page for help fixing common proble

Bugs

Head over to the bugs page for a list of known bugs.

RPi Model B 3D CAD files

Theses are various 3D CAD Versions in both RAR and ZIP.

- CATIA V5 RAR http://sdrv.ms/JqdhMb
- CATIA V5 ZIP http://sdrv.ms/LjyLGD
- ProE RAR http://sdrv.ms/KCv1hZ
- ProE ZIP http://sdrv.ms/KCvhxq
- STEP RAR http://sdrv.ms/KCvv7T
- STEP ZIP http://sdrv.ms/JMhv18
- SketchUp http://scc.jezmckean.com/item/581
- SketchUp8 http://sketchup.google.com/3dwarehouse/details?mid=327d6b1d8bd6130d6fbd6b70c7f1d3e0
- Eagle 5 http://www.raspberrypi.org/phpBB3/viewtopic.php?f=41&t=4457
- STEP and various formats at tracepartsonline
- http://www.tronetix.com/joomla/index.php/projects
- http://www.hycshop.com/raspberry-pi-c-2_17/
- Maya, Autodesk FBX, Wavefront ZIP http://www.raspberryconnect.com/idonethat/item/166-raspberry-model-download}

RPi Model B+ 3D CAD files

- STEP ZIP http://www.inti-innovations.co.uk/products/inti-media/i media-raspberrypi.html (Docs and downloads section)

Education Material

- The Raspberry Pi Education Manual(PDF) http://downloads.raspberrypi.org/Raspberry_Pi_Education_Man
- Raspberry Pi GPIO Pin Worksheet for free (re)use (Multi Forma http://raspberry.tips/raspberrypi-tutorials/raspberry-pi-modell-b-p gpio-pins-worksheet/

Books

- Heitz, Ryan. Hello! Raspberry Pi. Manning Publications (2015). pp.
    1. ISBN 9781617292453

Education Material

- The Raspberry Pi Education Manual(PDF) http://downloads.raspberrypi.org/Raspberry_Pi_Education_Manual.pdf

modell-b-plus-gpio-pins-worksheet/

Community

Projects, Guides & Tutorials

- An important source of information and guides is the Official Forum.
- Get started by following some of the many Tutorials.
- Common tasks and useful tips are available through the Guides page.
- Projects can be found, and added to, on the Projects page.
- Raspberry Pi Datasheets can be found on the DataSheets page.
- Knowledgeable users may want to review and help out with project wishlist items on the Tasks page.
- There are many tutorials, example projects and guides in The MagPi Magazine - which is available free online or to purchase in printed form.
- Some more great projects and setup guides in the Raspberry Pi Handbook

Schools, Universities, Clubs & Groups

- The Raspberry Pi Foundation's aims include encouraging education. Several groups including Computing At School aim to bring Computing Science back into schools.
- Go to the Education Page to add your project and find helpful links.
- Raspberry Jams are a great way to meet other Raspberry Pi users, share ideas and tips and learn more. To find a Raspberry Jam near you, see the Raspberry Jam page.

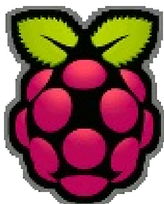
Supporting Communities

The Raspberry Pi Community is steadily growing:

- The Official Raspberry Pi Forum
- Element 14 Raspberry Pi Group, community site of Premier Farnell
- DesignSpark, community site of RS-Components
- 'Frambozenbier' (Raspberry Pi Homebrew)
- Stack Exchange Forum
- Non-official community of Raspberry Pi in spanish language
- World Of Pi A forum based on all things Raspberry Pi.
- The MagPi Magazine - Community based, free eMagazine, get involved!
- RaspberryPi Osdev - Hardware specific OS-development community, sitting in freenode.net#raspberrypi-osdev.
- news:comp.sys.raspberry-pi - Usenet newsgroup

About the RPi Wiki

Do not be afraid to add your bit, content is vital for the wiki to function.



A 3D rendering of the Raspberry Pi logo

Translations

The wiki is being translated into several languages, some of which can be seen on the hub banner above. Current languages include:

- English: R-Pi Hub
- Français: FR:R-Pi Hub

- Français: FR:R-Pi Hub
- Português: Pt-BR:R-Pi Hub
- 简体中文: Zh-CN:RPi_信息中心
- Deutsch: DE:R-Pi_Hub
- മലയാളം: Ml:R-Pi Hub

Any help translating would be greatly appreciated. Thank you to those who have already contributed!

References

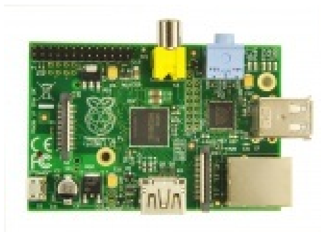
1. ↑ http://www.raspberrypi.org/archives/3195
2. ↑ http://www.element14.com/community/docs/DOC-44828/l/raspberry-pi-safety-data-sheet

3. v
4. t
5. e

Raspberry Pi

Startup

Buying Guide - SD Card Setup - Basic Setup - Advanced Setup - Beginners Guide - Troubleshooting



Hardware

Hardware - Hardware History - Low-level peripherals - Expansion Boards

Peripherals

Screens - Cases

- Other Peripherals (Keyboard, mouse, hub, wifi...)

Software

Software - Distributions - Kernel - Performance - Programming - VideoCore APIs - Utilities

Projects

Tutorials - Guides - Projects - Tasks - DataSheets - Education - Communities

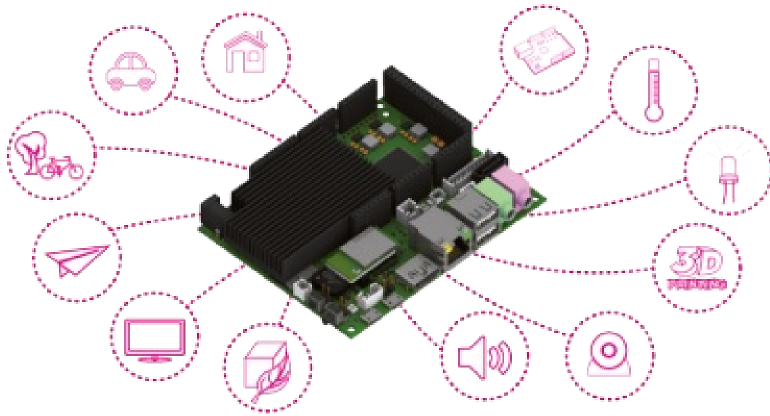
Category:

- RaspberryPi

From: eLinux.org

UDOO

This page collects information about UDOO the first ever small sized computer that merges in one single board an ARM cortex-A9 iMX.6 CPU and an Arduino Due compatible board embedded with a dedicated ARM SAM3X8E CPU. UDOO project has been funded through Kickstarter, raising $641,614 in sixty days thanks to 4,172 backers.



Notice: The UDOO Wiki pages on this site is collaborative work - the UDOO Team is **not** responsible for content on these pages.

Contents

What's UDOO?

UDOO is a single board computer that can be used both with Android and Linux, paired with an Arduino-compatible processor. It is a powerful prototyping board for software development and design; it's easy to use and allows developing projects with minimum knowledge of hardware design. UDOO merges different computing worlds together: each one has its proper strengths and weak points, but all of them are useful in todays life for educational purposes as well as Do-It-Yourself (DIY) and quick prototyping. UDOO is an open hardware, low-cost platform equipped with an ARM i.MX6 Freescale processor, and an Arduino Due compatible section based on ATMEL SAM3X8E ARM processor, all this available on the same board!

UDOO goals:

- Develop an innovative product for a growing market
- Give a new vision to the educational framework, with the idea of training up a new generation of engineers, designers and software developers skilled in digital technology: physical computing, multi-media arts, interactive arts, IoT...
- Give a boost to the DIY world
- Offer a low cost embedded platform for interactive arts with powerful tools: Processing, OpenCV, PureData, openFramework
- Provide companies with a great tool for fast prototyping

Specifications

UDOO retail line up consists of three models, sharing most of the features and different only for connectivity and i.MX6 processor used. All three models feature an embedded Arduino compatible section based on Arduino Due schematic. UDOO's dimensions are: 4.33 inch x 3.35 inch (11 cm x 8.5 cm).

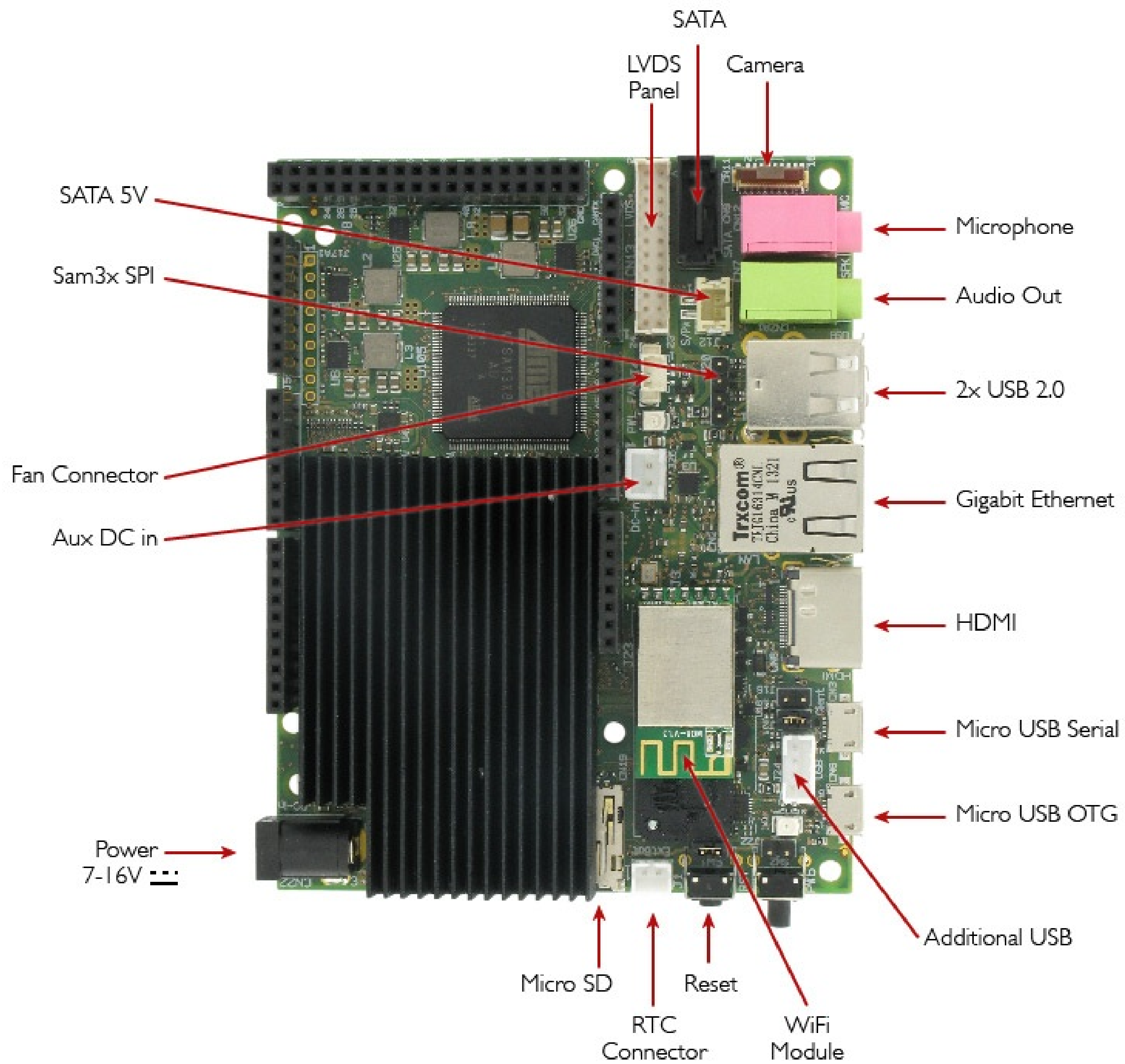
	UDOO Dual Basic Do it EASY	UDOO Dual Do it FLEXIBLE	UDOO Quad Do it POWERFUL
Model Images are for illustrative purposes only. Actual product may differ			
CPU	Freescale i.MX 6 DualLite + Atmel SAM3X8E	Freescale i.MX 6 DualLite + Atmel SAM3X8E	Freescale i.MX 6 Quad + Atmel SAM3X8E
GPU	Vivante GC 880 + Vivante GC 320	Vivante GC 880 + Vivante GC 320	Vivante GC 2000 + Vivante GC 355 + Vivante GC 320
Arduino compatible board	✓	✓	✓
76 Fully Available GPIO	✓	✓	✓
WIFI Module	✗	✓	✓
Gigabit Ethernet	✗	✓	✓
SATA Support	✗	✗	✓

- Freescale i.MX6Quad, 2\4 x ARM® Cortex™-A9 core @ 1GHz with ARMv7A instruction set
- GPU Vivante GC 2000 for 3D + Vivante GC 355 for 2D (vector graphics) + Vivante GC 320 for 2D
- Atmel SAM3X8E ARM Cortex-M3 CPU (same as Arduino Due)
- RAM DDR3 1GB
- 76 fully available GPIO with Arduino compatible R3 1.0 pinout
- HDMI and LVDS + Touch
- 2 Micro USB (1 OTG)
- 2 USB 2.0 type A and 1 USB 2.0 internal pin header (requires adapter cable)
- Analog Audio and Mic jacks
- CSI Camera Connection
- on board Micro SD card reader (boot device)
- Power Supply (6-15V) and External Battery connector
- Ethernet RJ45 (10/100/1000 MBit)
- WiFi Module
- SATA connector with power header

Warning: The UDOO I/O pins are 3.3V compliant. Higher voltages (like 5V) would damage the board.

Learn more about wrong uses that invalidate the warranty.

Board Overview



GPIO features

- 76 fully available GPIO
- Arduino-compatible R3 1.0 pinout
- 3,3 V Compliant
- Compatible with All Arduino Due Shields and most Arduino Shields
- GPIO's can be accessed as Arduino pins, GPIO's or as additional S\PDIF, FlexCAN,I2S, SPI

More informations about UDOO_GPIO_Pinout

Getting Started

Very first start

- An easy step by step guide that will lead you to boot your UDOO for the very first time.
- How to Create a bootable micro SD card for UDOO
- Configure your UDOO with UDOO Configuration Tool

Advanced Setup

- How to Create a bootable Micro SD card from precompiled binaries.
- How to Create a bootable Micro SD card from sources.
- How to Update UDOO Kernel
- How to boot from SATA drive
- Using USB Debug Connection
- **Having problems? Try the Troubleshooting page.**

Resources

UDOO has a very active and growing community where to find help and new ideas

- UDOO.org is the official Website
- UDOO Forum is a great place to start discussing
- Get started with some basic projects and tutorials:
  - UDOO YouTube Tutorials
  - UDOO Tutorials Section
  - UDOO Projects Section
- Take a look at UDOO User Manual which contains lots of useful technical informations
- Hop on UDOO Channel IRC Chat

Tutorials

Linux

- How to install a custom Debian distro with debootstrap
- How to Create a Virtual Machine for UDOO Development
- Understand some basic linux commands

Android

- Introduction on Making with Android
- How to compile android from sources
- How to Switch between adb Debug and ADK connection
- A useful Android ADK Toolkit Library and its Docs
- How to configure Ethernet under Android

How to configure Ethernet under Android

Arduino

- How to program the embedded Arduino microcontroller

Resources

Hardware & Accessories	Software & OS Distributions	Additional Resources
• UDOO_GPIO_Pinout • IMX 6 Internal and drivers pin-muxing reference • IMX6 and Sam3X Communication • How to Setup LVDS Display Panels • How to setup UDOO Camera Module • Using Watchdog Timer on UDOO	• UDOOBuntu is the Official UDOO Linux Distribution • Android 4.3 is the Official UDOO Android Distribution • List of allUDOO Distributions available	• Node-udoo is an abstraction library for Node.js complete with command line tools (callback, promise, and synchronous styles supported) ◦ Introductory/demo video ◦ Project homepage ◦ Installing the latest node.js is covered in this forum post

Official Accessories

UDOO Camera Module

- Auto focus control (AFC) with embedded AF VCM driver
- Sensitivity: 600mV/lux-sec
- Video capture in Full Field of View (FOV): double sensitivity,improved signal-to.noise ratio (SNR)
- Post-binning re-sampling filter for sharper, crisper contours and colours
- Internal anti-shaking engine
- Image transfer rate

VGA (320x480) @120fps VGA (640x480) @90fps 720p @60fps 1280x960 @45fps 1080p @30fps QSXGA (2592x1944) @15fps

More informations about UDOO Camera Modules

UDOO Camera Module Datasheet



UDOO LVDS Touch Screens

7" Touch Panel Kit

- 7" TFT RGB Display
- I2C Touch Screen
- Dual Touch
- Resolution 800X480
- UDOO_VK-7T video cable for UDOO
- LCD BOARD ADAPTER

How to setup lvds panels

UDOO 7" Touch Panel Display Kit Datasheet



15" Touch Panel Kit

<table>
<tr>
<td>

- 15,6" LVDS Display
- USB Capacitive Touch Screen
- Resolution 1366X768 24bit
- UDOO_VK-15T video cable for UDOO
- USB CABLE for Third UDOO'USB
- Touch Controller Board

How to setup lvds panels

UDOO 15" Touch Panel Display Kit Datasheet

</td>
<td></td>
</tr>
</table>

Community

Forums The official UDOO forums can be found at http://www.udoo.org/forum

The forum search facility has been tweaked to allow more general searching. **Please** do a search before making a post as the issue may already have been raised and answered.

IRC There is an (unofficial) UDOO discussion channel on IRC. Using the IRC client of your choice, use server information: irc.freenode.net. Room name is #udoo.

Home site and community

1. Official web site http://www.udoo.org
2. Official forum http://www.udoo.org/forum/index.php

Social account

1. Facebook fan page http://www.facebook.com/udooboard
2. Twitter http://twitter.com/UDOO_Board
3. Google+ https://plus.google.com/u/0/110742692974455430878/posts
4. YouTube http://www.youtube.com/channel/UCXv5UyGn5jArK8xOAmuSeHg

Category:

- UDOO

> From: eLinux.org

Improv

The Improv is a modular engineering kit that comes with a CPU card and a separate feature card to connect it to. The CPU card powers the device while the feature card provides convenient access to the on-board capabilities of the CPU card through a variety of ports and pins on it.

This unique design allows one to have multiple CPU cards for different projects, upgrade the kit by ordering new CPU cards for additional features or feature cards for different sets of ports and off-board components.

Improv comes pre-installed with the Linux-based Mer operating system. We chose Mer because it has been designed specifically for device development, has a thriving community of both companies and enthusiasts and has a proven open governance model directing development. You aren't locked into this, however: you can install the operating system of your choice.

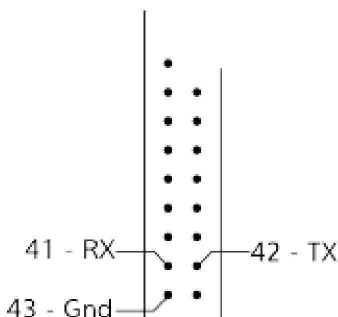
Contents

Hardware

Availability

I/O interfaces

Serial connection on the 44 pin DIL



Ubication of the pin for a uart serial connection on the 44 pin DIL of an improv board

With an usb-serial adaptor, it's possible to connect to the Improv with software such as Minicom to obtain a serial console with debug messages and a terminal.

To configure minicom to connect, simply do:

- Run minicom: minicom -s
- Scroll down to Serial port setup and hit enter
- Select option A to set the serial device to /dev/ttyUSB0 and then hit return.
- Select option E and change to 115200 N81 to set speed, parity, number of bits and number of stop bits.
- Select option F and set to No to indicate no hardware flow control is needed.
- Select option G and set to No to indicate no software flow control is not required.
- Hit return again to leave the menu.
- Scroll down to Exit and hit enter - this will exit the menu and start up the minicom software.

Operating system

Improv comes with a Linux distribution based upon the Mer operating system preinstalled. The full image can be found [here http://makeplaylive.com/files/improv-images/plasma-active-armv7hl-sunxi-eoma68-improv-nand-latest.raw.bz2] and can be re-flashed any time on the device.

On this site, there are several ready to use, nand and microsd based images, that are described here. On the same Linux-sunxi wiki page, there is described the procedure on how to generate operating system images based upon Mer and Plasma Active.

It's possible to install any different operating system on the board, or to boot from the microSD card.

On the Linux-sunxi wiki, you can find a tutorial on installing a generic operating system in the nand, from a system running on top of a microSD.

Recovery

Links

> From: eLinux.org

OpenPhoenux

Contents

Welcome to the OpenPhoenux Overview Page



OpenPhoenux Logo

This is a place where OpenPhoenux fans can come to find information on the OpenPhoenux GTA04 and OpenPhoenux Neo900 projects, which are created by Goldelico. Most of this information can also be found on the OpenPhoenux.org site and is provided here as a convenience and a possibility to collaborate with a broader community.

"The goal of the OpenPhoenux project is to provide a complete community driven and independently developed hardware, software and services platform for portable operation, that is as free and open as we can achieve it within our environment. This includes kernel, drivers and application software a well as documented hardware, and open discussion. It continues where the Openmoko project did end." (from the projects Manifest)

Interested people are invited to **join the discussion** at: http://lists.openphoenux.org/mailman/listinfo/community

OpenPhoenux GTA04 Project

The GTA04 is a motherboard, which functions as a base-board for different devices. It was originally designed as an upgrade/replacement board for the Openmoko Freerunner.

Hardware

There are different revisions of the GTA04-Board, which have slightly different technical specifications.

Devices

- Letux 2804 (Smartphone)
- Letux 3704 (Professional PDA)
- Letux 7004 (Tablet)

Ressources

http://www.OpenPhoenux.org

OpenPhoenux Neo900 Project

The Neo900 project provides an upgrade/replacement motherboard for the Nokia N900 and also fully assembeled, pre-upgraded Neo900 devices.

Hardware

The Neo900s board is roughly based on the GTA04s design.

Devices

- Neo900 (Smartphone)

Ressources

http://www.Neo900.org

Related Projects

- Openmoko: http://www.openmoko.org
- OpenPandora: http://www.openpandora.org
- Dragonbox Pyra: http://www.pyra-handheld.org
- QtMoko: http://qtmoko.sf.net
- Replicant: http://www.replicant.us
- SHR-Project: http://www.shr-project.org
- FSO: http://www.freesmartphone.org
- PowerVR SGX FOSS driver: http://powervr.gnu.org.ve

Category:

- OpenPhoenux

> From: eLinux.org

Jetson TK1

Contents

About this site

This is the official Wiki for embedded Tegra & the Jetson TK1 board, maintained by both the community and NVIDIA.

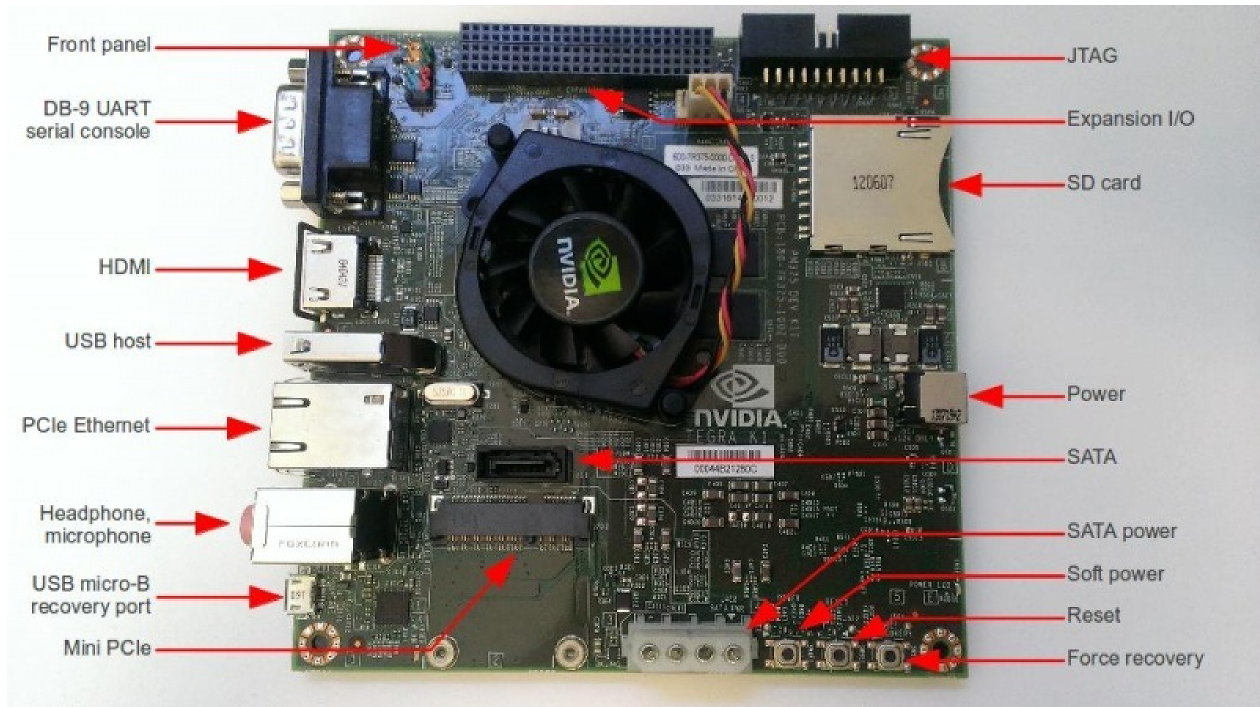
The other embedded Tegra community sites with official NVIDIA support are:

- The forum for discussing embedded Tegra & Jetson TK1 issues with the community & NVIDIA.
- The blog to stay updated with the latest news & plans for embedded Tegra & Jetson TK1 from NVIDIA.

About Tegra K1

Tegra K1 is NVIDIA's first mobile processor to have the same advanced features & architecture as a modern desktop GPU while still using the low power draw of a mobile chip. The Jetson TK1 board therefore allows embedded devices to use the exact same CUDA code that would also run on a desktop GPU (used by over 100,000 developers), with similar levels of GPU-accelerated performance as a desktop.

About Jetson TK1



Jetson TK1 is NVIDIA's embedded Linux development platform featuring a Tegra K1 SOC (CPU+GPU+ISP in a single chip), selling for $192 USD. Jetson TK1 comes pre-installed with Linux4Tegra OS (basically Ubuntu 14.04 with pre-configured drivers). There is also some official support for running other distributions using the mainline kernel, discussed further in the Distributions and Mainline kernel sections below.

Besides the quad-core 2.3GHz ARM Cortex-A15 CPU and the revolutionary Tegra K1 GPU, the Jetson TK1 board includes similar features as a Raspberry Pi but also some PC-oriented features such as SATA, mini-PCIe and a fan to allow continuous operation under heavy workloads:

Hardware Features

- **Dimensions:** 5" x 5" (127mm x 127mm) board
- Tegra K1 SOC (CPU+GPU+ISP in a single chip, with typical power consumption between 1 to 5 Watts):
  - **GPU:** NVIDIA Kepler "GK20a" GPU with 192 SM3.2 CUDA cores (upto 326 GFLOPS)
  - **CPU:** NVIDIA "4-Plus-1" 2.32GHz ARM quad-core Cortex-A15 CPU with Cortex-A15 battery-saving shadow-core
- **DRAM:** 2GB DDR3L 933MHz EMC x16 using 64-bit data width
- **Storage:** 16GB fast eMMC 4.51 (routed to SDMMC4)
- **mini-PCIe:** a half-height single-lane PEX slot (such as for Wifi, SSD RAID, FireWire or Ethernet addon cards)
- **SD/MMC card:** a full-size slot (routed to SDMMC3)
- **USB 3.0:** a full-size Type-A female socket
- **USB 2.0:** a micro-AB female socket (for connecting to a PC, but can also be used as a spare USB 2.0 port using a micro-B male to female Type-A adapter that is sometimes included)
- **HDMI:** a full-size port
- **RS232:** a full-size DB9 serial port (routed to UART4)
- **Audio:** an ALC5639 Realtek HD Audio codec with Mic in and Line out jacks (routed to DAP2)
- **Ethernet:** a RTL8111GS Realtek 10/100/1000Base-T Gigabit LAN port using PEX
- **SATA:** a full-size port that supports 2.5" and 3.5" disks, but is not hot-pluggable. (Turn off the power before plugging in

SATA disk drives)
- **JTAG:** a 2x10-pin 0.1" port for professional debugging
- **Power:** a 12V DC barrel power jack and a 4-pin PC IDE power connector, using AS3722 PMIC
- **Fan:** a fan-heatsink running on 12V (to allow safely running intense workloads continuously, but can usually be replaced by a heat-spreader or heatsink)

The following signals are available through the 125-pin 2mm-pitch expansion port:

- **Camera ports:** 2 fast CSI-2 MIPI camera ports (one 4-lane and one 1-lane)
- **LCD port:** LVDS and eDP Display Panel
- **Touchscreen ports:** Touch SPI 1 x 4-lane + 1 x 1-lane CSI-2
- **UART**
- **HSIC**
- **I2C:** 3 ports
- **GPIO:** 7 x GPIO pins (1.8V). Camera CSI pins can also be used for extra GPIO if you don't use both cameras.

Front panel connector:

- Green - Power LED
- Orange - HDD LED
- Red - Power Button
- Purple / Blue - Reset Button

Hardware-accelerated APIs supported:

- **CUDA** 6.0 (SM3.2, roughly the same as desktop SM3.5)
- **OpenGL** 4.4
- **OpenGL ES** 3.1
- **OpenMAX IL multimedia codec** including H.264, VC-1 and VP8 through Gstreamer
- **NPP** (CUDA optimized NVIDIA Performance Primitives)
- **OpenCV4Tegra** (NEON + GLSL + quad-core CPU optimizations)
- **VisionWorks**

Buying Guide

Where can I get one and for how much?

- Visit the NVIDIA store then choose a distributor in USA, or click on "International Orders" to see other countries it is available from including UK, Germany, France, Italy, Japan, Russia, China, Singapore, and Australia.
- Jetson TK1 costs $192 in USA.
- The package includes a power supply (with a detachable US mains cord), a USB micro-B cable for connecting it to a PC, and usually a USB micro-B to female USB-A adapter allowing you to have 2 regular USB ports.
- There are a large number of optional accessories you can add to Jetson such as Cameras, SATA hard-disks and mini-PCIe devices.

Setting up a new board

To configure a new board, or factory reset an existing Jetson, visit the official Get Started On Jetson page, particularly to read the Quick Start guides to "flash" your device (wipe it clean and install Linux onto it). And if you will want to do software development for your Jetson, such as to build CUDA code, you should install the JetPack (Jetson Development Package) as mentioned in the Quick Start Guides to install a graphical IDE with cross-compilation, debugging & visual profiling tools.

Basic setup steps to access the board and access internet

Whether you want to sew your embedded Tegra into a backpack or put it in a robot or simply use it as an ultra powerful media center, the first thing you should do with a new Jetson TK1 board is attach it to a HDMI monitor & keyboard & mouse to make sure it works and get familiar with it for a few minutes.

Note: the micro-USB port on Jetson TK1 can be used as a second USB port if you use the supplied adapter.

The device can be accessed in 2 possible ways, depending on whether you want to plug a keyboard & mouse & monitor directly into the Jetson TK1 board or you want to plug an Ethernet cable between your device and a PC or laptop or router and access it through a network:

Direct access to a Jetson board using its own keyboard & mouse & monitor

The Jetson TK1 Quick Start Guide (included as a booklet with your Jetson TK1) shows how to use the Jetson TK1 board as a mini standalone computer. Basically, you plug in a HDMI monitor or TV, plug a keyboard into the USB3.0 port, plug a mouse into the included micro-B to female USB adapter and plug that into the micro-B USB2.0 port on the board. Then plug the 12V power supply in, press the small POWER button, then watch it boot up into Linux4Tegra (Ubuntu 14.04 with some drivers pre-configured). When it asks for the password for user "ubuntu", just type "ubuntu" to log in. If you have an Ethernet router then simply plug an Ethernet cable from the board into your router (or plug in a USB Wifi dongle) to have internet access, and you are ready.

Remote access to a Jetson board through the Ethernet port

To access the board remotely through a local network from a PC or laptop, follow the Remote Access instructions so you can control the device from the keyboard & mouse & monitor on your PC or laptop and share your desktop's Wifi or Ethernet internet access to the attached device.

An important step before connecting the Jetson to Internet

It is really important to tell "apt" not to overwrite the file "libglx.so" if you upgrade the system. "libglx.so" is a specific file in NVIDIA's graphics driver that might get replaced by an incorrect version from Ubuntu that stops you from being able to boot into the graphical environment! So please execute this command on your Jetson before you connect it to Internet or perform an update:

```
sudo apt-mark hold xserver-xorg-core
```

Now you can allow Ubuntu to update itself automatically or you can run "sudo apt-get upgrade" without problems. You can get more information about this issue on the official development forum. Note: Is this only a problem for L4T Rel-19 only? It seems fixed in Rel-21.

Recommended first steps now that your board has internet access

Open a command-line terminal to perform some initial operations. If you are using the Jetson TK1's graphical environment then click on the top-left icon in Ubuntu Unity and type "terminal" to open a command shell, or if you have remote access to the device through a network then open an SSH command shell into your device from your desktop such as by running "ssh ubuntu@tegra-ubuntu". Now you are ready for initial configuration.

Add the Universe package repositories, since you will often need packages from Universe for code development:

```
sudo apt-add-repository universe
sudo apt-get update
```

If you will use the shell command-line a lot

Install "bash-completion" (it allows you to hit the "Tab" key to auto-complete your shell commands) and "command-not-found" (it shows which package you probably need to install if you run an unavailable command). These 2 tools are extremely useful when using the commandline, but were not installed by default in Ubuntu 14.04. Simply run this:

```
sudo apt-get install bash-completion command-not-found
exit
```

Note: now you need to log back in for it to start using bash-completion and command-not-found.

You probably should also change the shell prompt (by adjusting "PS1" in the ".bashrc" file in your home directory) to be more useful, such as getting the shell prompt to have a different color than regular commands, and make it obvious if a command returned with an error. There are thousands of custom .bashrc configurations on the web, including Shervin's that provides a different colored shell prompt depending on whether a command was succesful or returned an error.

If you will use the graphical environment (Unity) a lot

You might want to try some suggestions at "http://itsfoss.com/things-to-do-after-installing-ubuntu-14-04/" or similar, such as to turn off the desktop shopping suggestions that are enabled by default in Ubuntu 14.04 (despite the spyware concerns discussed by huge numbers of people) by running this:

```
gsettings set com.canonical.Unity.Lenses disabled-scopes "['more_suggestions-amazon.scope', \
    'more_suggestions-u1ms.scope', 'more_suggestions-populartracks.scope', 'music-musicstore.scope', \
    'more_suggestions-ebay.scope', 'more_suggestions-ubuntushop.scope', 'more_suggestions-skimlinks.scope']"
```

If you need more disk space on the eMMC

The eMMC on the Jetson has a capacity of 16GB, however some instructions or boards default to only using 8GB of the drive (or 12GB, see discussion). If you require more disk space in your rootfs such as for installing toolkits or compiling large projects, you can flash the Jetson (from a Linux desktop) to have a larger filesystem (note that this will erase all data on the Jetson TK1, and it takes roughly 1 hour to flash the whole eMMC!):

```
[user@host Linux_for_Tegra]$ sudo ./flash.sh -S 14580MiB jetson-tk1 mmcblk0p1
...
sending file: system.img
/ 15032385536/15032385536 bytes sent
system.img sent successfully
...
Create, format and download  took 2791 Secs
Time taken for flashing 2792 Secs
```

The maximum value of the `flash.sh -S` flag that works successfully is 14580MiB. Greater values (like 16GiB) are beyond the capacity of the eMMC when the other system-required partitions are included.

More recommended Linux tips for a Jetson TK1

Some more tips (such as how to speed up SSH logins, login automatically, share your keyboard & mouse, etc, are in a forum discussion.

Shutting down Jetson TK1 safely

Just like any Linux computer, the recommended way to shut-down or turn off Jetson TK1 is to click Shutdown in the GUI or run this in a terminal, to ensure the filesystem will not be corrupt:

```
sudo shutdown -h now
```

Emulators for developing with Jetson TK1

- **Emulators:** which Windows, OSX, Linux or SteamOS products with a powerful Nvidia GPU are economic and suitable for developing Jetson software?

Books

Explore your Jetson TK1 (e-book)

The JetPack Cookbook (proposed)

Tutorials for developing with Jetson TK1

The following are tutorial projects for hardware and/or software development. They show the easiest way to do a certain task, while the Reference Information section below has more detailed pages. If you have something useful to contribute about Jetson TK1 or embedded Tegra then please do so.

Tutorial

Description

Programming Difficulty

Electronics Difficulty

Hello World

Create a simple program that prints "Hello World!", by compiling code directly on your device

#

CUDA

Install CUDA then build & run some CUDA sample projects

#

Nsight

Documentation and tutorials on GPU debugging and profiling with Nsight, which is installed with CUDA

#

OpenCV

Install OpenCV then grab camera frames or build & run some OpenCV samples

#

OpenGL

Links to many OpenGL and OpenGL ES sample projects

#

Full Body Detection

Shows how to perform full body detection (something that even desktop CPUs are too slow for!) from a webcam and display the face, using OpenCV

#

GPIO Input & Output

Turn on an LED or send a signal to an Arduino microcontroller using GPIO

#

#

Vision-controlled GPIO

Turn on an LED whenever a face is detected in your camera

#

#

Battery Power

Power your Jetson TK1 from a battery pack

#

Automatic Pan Tilt

Build an autonomous pan-tilt face tracking camera, that tracks faces as they move around

#

#

Optical Flow Motion

Generate the optical flow motion vectors to see how things are moving

##

Video Stabilization

Real-time video stabilization such as for a robot's onboard camera

##

Program An Arduino

Program an Arduino microcontroller from your Jetson TK1 (instead of from a PC)

##

Communicate To An Arduino

Connect an Arduino board to a Jetson TK1 board, with communication between then

##

##

Follower Robot

Get a wheeled robot to drive towards the nearest person

##

##

Walking Follower Robot

Get a 2-legged robot to walk towards the nearest person

###

##

Jetson TK1 Reference Information (more detailed than the tutorials above)

General Topics

- **Official Jetson TK1 website**.
- **Official Jetson TK1 software development page**, including L4T and all the Linux kernel source changes.
- **System Info:** Find out the hardware info & what is available on the Jetson TK1 such as disk space, RAM, and devices.
- **Cloning & Backup** Save the Jetson's eMMC (system.img) via the flash port and restore it to other Jetson boards.
- **Trace32 / JTAG Debugging** Attaching a professional debugger module for very low-level access to the Tegra.

Networking Topics

- **Wifi & Ethernet Adapters:** Discusses which Wifi or Ethernet cards have been tested on Jetson TK1.
- **Remote Access:** Control the device from your PC or laptop, and share your desktop's Wifi or Ethernet internet access to the attached device.

Software Topics

- **CUDA Installation:** How to install the CUDA toolkit.
- **VisionWorks Installation:** How to install the VisionWorks toolkit.
- **OpenCV Installation:** How to install OpenCV, including building from source.
- **ArrayFire Installation:** How to install ArrayFire for Tegra and getting started.
- **Kodi (XBMC) Installation:** "Jetson/Kodi (XBMC)") How to install XBMC Media Center such as for a green HTPC.
- **Plex Media Server Installation:** How to install Plex Media Server such as for a green HTPC.
- **Libraries:** Libraries that have been tested on Jetson TK1.
- **Web Browsers:** How to get web browsers working, including Flash for watching youtube videos in Chromium.
- **H.264 Codec:** Hardware-accelerated video encoder/decoder for H.264 and other multimedia formats.
- **cuDNN:** GPU-accelerated Machine Learning & Deep Neural Network library.
- **JavaFX installation:** How to install JavaFX for the Jetson TK1

Performance and Power Topics

- **Performance:** How to maximize or minimize CPU & GPU clocks for different performance & power draw.
- **Power Management:** Electrical power related issues, such as powering the board from a battery or seeing how much power is drawn.
- **Thermal:** How to replace the fan with a heat-spreader or heatsink, and measure the board's temperatures.
- **Graphics Performance:** Discusses graphics performance of the Tegra TK1 on the board, including comparison with competing solutions.
- **Computer Vision Performance & Power Draw:** Describes the different Tegra hardware optimizations in OpenCV, and shows power & perf measurements for many computer vision applications in OpenCV, CUDA and VisionWorks.

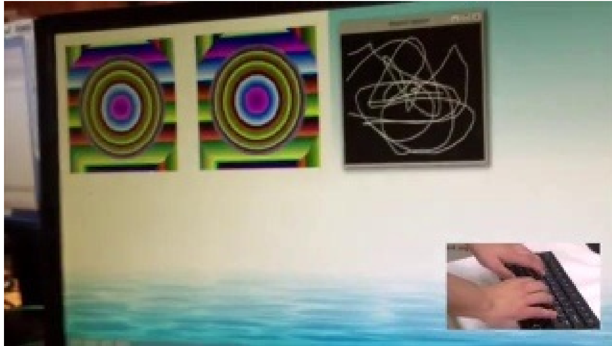
Hardware Expansion Topics

- **Cameras & Webcams:** Discusses USB cameras and CSI cameras, stereo cameras and Time-Of-Flight 3D depth cameras.
- **mini-PCIe Add-on Cards:** Shows some possible add-on cards that can be used in the mini-PCIe port.
- **GPIO:** How to control digital output pins and read input data pins on Jetson TK1.
- **PWM:** How to control the spped of motors & servos or LED brightness using Jetson TK1.
- **I2C:** How to communicate between microcontrollers & devices using I2C on Jetson TK1.
- **Bluetooth:** Use wireless Bluetooth devices such as keyboards, mice & speakers, or communicate to a smartphone or tablet.
- **Enclosures & Cases:** Various cases that are available to protect a Jetson TK1.

- **Audio:** Issues related to audio input or output, such as microphones or speakers.
- **RTC:** Add battery backup RTC and support it using the Grinch Custom Kernel.

Projects using Jetson TK1

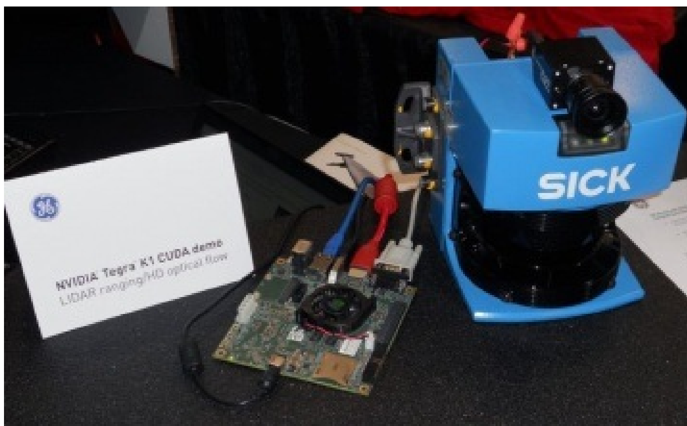
The following are descriptions, photos and/or videos of projects featuring Jetson TK1. Feel free to add your own!



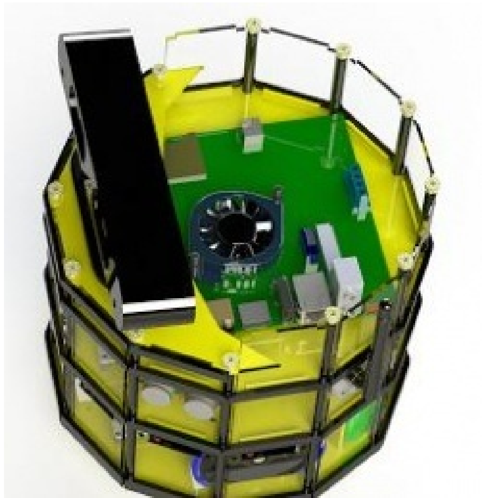
Upstream Linux 3.15 + Wayland + Weston + Nouveau running a 100% open-source OS on Jetson TK1, by CodeThink in UK. If you want to try a similar setup on your Jetson TK1, read this and this.



NVIDIA's SCOL "Super-Computer-On-Legs" robot, by Shervin Emami in Australia. Performs Optical Flow video stabilization and HOG person detection to walk towards nearby people. Operator Manual.



Jetson TK1 Lidar range finder + camera Optical Flow robotics demo by General Electric Intelligent Platforms in USA. GE IP are developing a ruggedized Tegra K1 based module.



Explorer Robot by the Officine Robotiche in Italy

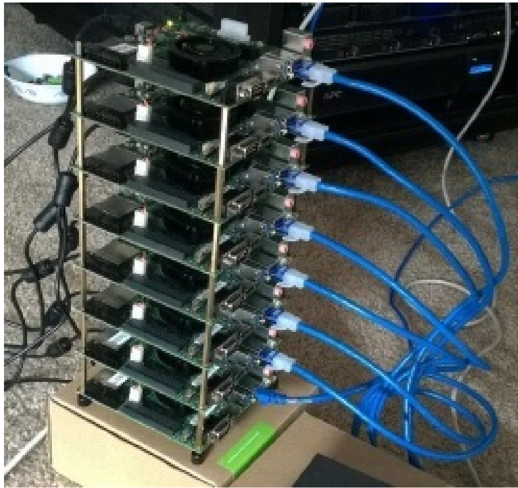


_1426607684475.jpg "MyzharBot Robot")

MyzharBot by Walter Lucetti in Italy is a crawler robot made to study Navigation algorithms based on Computer Vision, Machine Learning and 3D Sensors. The project is open source and open hardware and has its own blog where you can find every information to replicate it. MyzharBot has partecipated to GTC2015 conference running for 3 days between the legs of the attendants of the Nvidia's booth in the exhibit area.

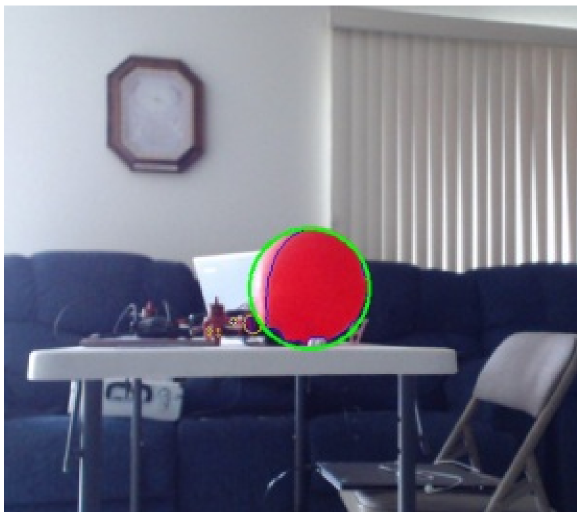


Project TURBO (TK1 Unmanned Reconnaissance Bot) is a low-cost mobile research platform developed by GE Intelligent Platforms, exploring CUDA-accelerated autonomy, sensing & perception powered by Tegra K1.

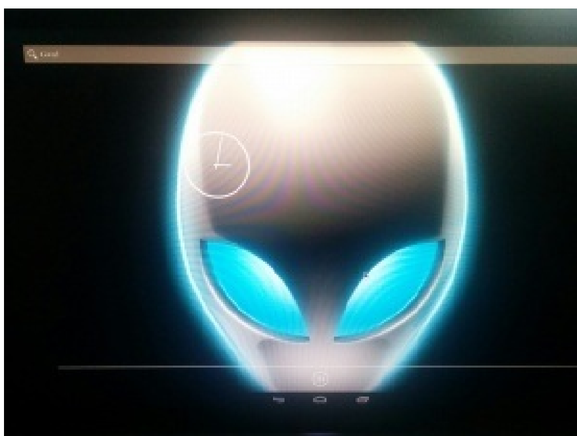


Several users are building cluster computers using a handful of Jetson TK1's:

- User rralf is using 10-15 Jetson TK1's.
- User Purex is using 8 Jetson TK1's.



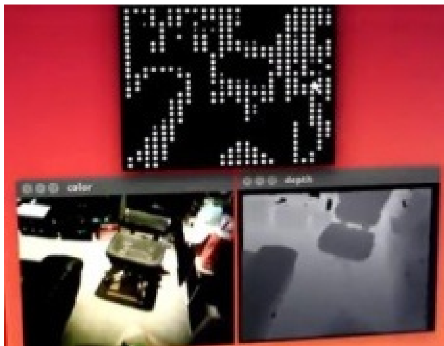
A user is using CUDA accelerated computer vision to find balloons (or other red round things). This article explains the algorithm and has a link to the source.



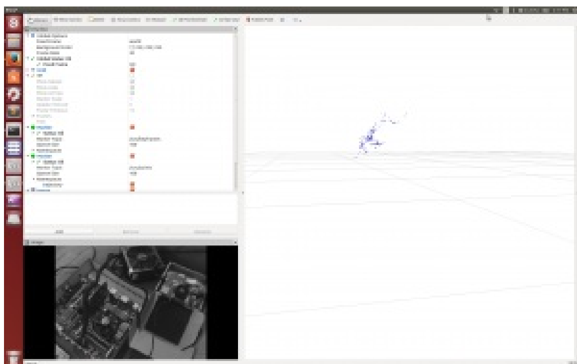
Project Jedroid Android on Jetson TK1. Pure AOSP, Pure open source code. Jedroid source code now avaiable, let's work on jedroid together



NBA 2K14 Android game using Jedroid on JetsonTK1 Demo video



TR4Tegra. Smart-glasses with a depth camera, connected to Jetson TK1 in a backpack, detects object location and notifies a blind user with a ~25x25 neural implant info



Monocular Visual Odometry for UAV using ROS on JETSON TK1 Project Info



A low cost vision based development platform based on the Nvidia Jetson TK1. Project info



Jetson TK1 running on Android Lollipop BSP from e-con Systems with MIPI camera and Ethernet connectivity. Demo video Project info

Linux distributions running on Tegra

Jetson TK1 comes preloaded with NVIDIA's Linux4Tegra (L4T) distribution. However it is possible to install other distributions on a Tegra device:

- Linux4Tegra (L4T) distribution: basically Ubuntu 14.04 with pre-configured drivers for bootloader, kernel, OpenGL, X.Org, Multimedia, etc.
- Gentoo Linux from SD: run Gentoo directly on an SD-card without modifying the contents of the eMMC storage. (Will soon support many other Linux distros such as Debian, Ubuntu, Redhat, etc!)
- Gentoo Linux: run Gentoo Linux (from eMMC) instead of Ubuntu Linux.
- Busybox: create a very minimal root filesystem with Busybox.
- Android: The user "lucasdai" is working on a community-supported version of Android on Jetson TK1.
- Android lollipop on Jetson TK1: Android Lollipop port for Jetson TK1, inlcuindg support from e-Con Systems.
- others?

Upstream/mainline OS & kernel system software on Tegra

NVIDIA's Tegra SoCs are well supported by mainline OSS such as Linux and U-Boot. This section contains an index of topics related to running mainline software on Tegra.

- U-Boot.
- Linux kernel.
- Nouveau driver.
- Issues specific to running mainline Linux on the Jetson TK1 board.

Jetson TK1 and Tegra K1 Hardware Documents

The official Jetson TK1 **Hardware Design and Development page** contains the open-source hardware documents, allowing you to inspect your Jetson board or even design your own custom board based on Tegra K1 or Jetson TK1, including:

- Jetson TK1 DevKit Specification
- Jetson TK1 Schematics
- Jetson TK1 PCB Board Files (Gerbers)
- Jetson TK1 2D CAD file with layers (.DXF)
- Jetson TK1 Bill of Materials (BOM)
- Tegra K1 SOC Technical Reference Manual (TRM)
- Tegra K1 Embedded Platform Design Guide
- Tegra K1 Memory Approved Vendor List
- Jetson TK1 Pin Mux
- Jetson TK1 Allegro Design File
- Jetson TK1 PCB Stack Up Details
- Jetson TK1 ValorODB++ Database
- Jetson TK1 Datasheet Orcad Schematics
- Jetson TK1 Board Orcad Schematics
- Jetson TK1 PCB Assembly Drawing
- Jetson TK1 RS274x Gerber Data
- Jetson TK1 Autocad DXF Format Design File
- Jetson TK1 PCB Mentor PADS ASCII Format Layout

Other embedded Tegra boards created by NVIDIA

Jetson TK1 was the first embedded board that NVIDIA created for the general public, but there have also been some other Tegra boards, including the automotive-grade Tegra-K1 based Visual Compute Module and the Jetson Pro development platform, both for the automotive industry (requires an NDA and large sales figures, etc).

Other SOCs created by NVIDIA

SOCs: NVIDIA made several previous generations of Tegra SOCs for the mobile, automotive and MP3 player industries.

Other links

Resources is a list of links to Tegra-related documentation and code outside of this wiki, such as TRMs and mainline kernel links.

Google+

References

Categories:

- NVIDIA
- Tegra
- Jetson

> From: eLinux.org

Tegra

This page aims to be the primary reference/index for information on NVIDIA's Tegra SoCs, the Linux-based software stacks that run on Tegra, and related tools and techniques.

Sub-pages contain information on:

- The Tegra SoCs themselves.
- Boards containing Tegra.
- Running Mainline/Upstream software on Tegra. For example, the main Linux kenel or U-Boot.
- Running Downstream software on Tegra. For example, Linux4Tegra or L4T.
- Links to external resources.

Categories:

- NVIDIA
- Tegra

> From: eLinux.org

Parallella

Contents

NEWS:

- 6th May 2014: 100,000+ cores shipped!.

About

The Parallella project was inspired by hardware communities such as Raspberry Pi, BeagleBoard and Arduino, and aims to democratise access to parallel computing through providing an affordable open hardware platform and open source tools, and supporting learning and the development of software which is able to harness the power of parallel systems.

Motivation

Making parallel computing easy to use has been described as being *a problem as hard as any that computer science has faced*. With such a big challenge ahead we need to make sure as many people as possible have access to open parallel hardware and development tools.

Principles

The Parallella platform is being built on the following principles:

- Open Access: all architecture and SDK documents openly published on the Web — no NDAs or special access required.
- Open Source: software platform based on free and open source software (F/OSS) development tools and libraries. Board design files provided under an open source hardware license once the Parallella computer ships.
- Affordable: the goal is to bring the hardware cost to below $100, making it an affordable platform for all.

Hardware

See Parallella Hardware.

Software

See Parallella Software.

- v
- t
- e

Parallella

Startup

Ordering - Cases and Cooling - Quick Start Guide - SD Card Setup Macintosh



Hardware

Hardware - Daughter Cards

Peripherals

Cases - Peripherals

Software

Software - SDKs

Projects

Tutorials

- Guides - Projects

Category:

- Parallella

> From: eLinux.org

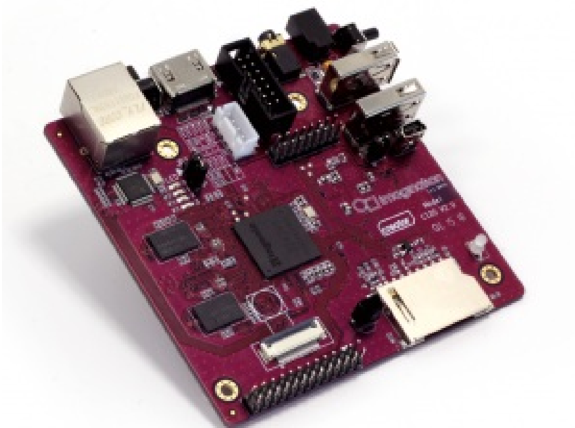
MIPS Creator CI20



The Ci20 board (V1)



The Ci20 board (V2)



The Ci20 board (V3)

The MIPS Creator CI20 platform is a feature laden MIPS/Imagination Linux and Android development system. It incorporates an Ingenic JZ4780 SoC which includes a 1.2GHz dual core MIPS32 processor and Imagination PowerVR SGX540 GPU. The CI20 board provides comprehensive connectivity, multimedia capabilities and substantial RAM and flash. CI20 is preloaded with Debian7, and other distros are being packaged to be available for download soon.

CI20 is an open platform with technical manuals, schematics and source code freely downloadable.

CI20 is now available to order from the Imagination Technologies store for £50 or $65.

Getting Started

Beginners Guide

- Which helps answer the questions:
  - Do I just plug it in?
  - What can I do with it out of the box?
  - What can I plug in and where?
  - Can I update the software?
  - Where can I get help?
  - Headless setup
- Troubleshooting guide

Technical Stuff

- What are the specs of the board?
- What SoC does the board use?
- Where can I find board schematics and SoC documentation?

Download Page

- Where do I get the documentation?
- Can I get the schematic?
- Where do I get new software from?
- Is there source code?

More Stuff

Tutorials

- Blink an LED

Projects

- Who is doing what with the CI20
- Ideas for what you can do with the CI20
- ToDo list

Distros and Images

- What different distros/software are available for CI20

Dev Zone

- Deeper technical details
- How and where do I contribute
- Where to find Bug trackers, Mailing lists and Forums, IRC.
- Hardware Testing
- Upstream
- ToDo and things to fix or improve

External Links

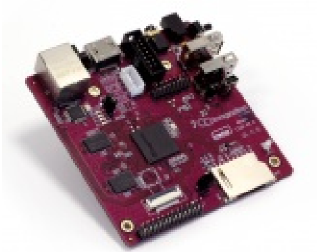
- Original Announcement (blog.imgtec.com)
- "MIPS Creator CI20 development board now available for only $65" (blog.imgtec.com)
- Imagination Technologies store (store.imgtec.com)

- v
- t
- e

MIPS Creator Ci20

Startup

How to get one - Beginners Guide - Tutorials - Downloads - Troubleshooting



Hardware

Hardware - Specs - Documentation

Software

Distributions - Debian - Android - Gentoo - Yocto - Ångström - Arch - OpenWRT - Buildroot

DevZone

DevZone - Source Code - Upstream - Mailing Lists - IRC

Projects

Projects

Category:

- CI20

> From: eLinux.org

Banana Pi

The Banana Pi is an open source computer capable of running Linux and Android 4.4. It has an AllWinner A20 SoC with 1GB of memory and a dual core ARM CPU.

Contents

Linux on Allwinner SoC resources

- Linux Sunxi Wiki

Banana Pi Resources

- Downloads
- Quickstart
- Banana Pi tutorials and usage information

Banana Pi Community

- LeMaker's Banana Pi forum
- Banana Pi Wiki

Specifications

CPU	A20 ARM® Cortex™-A7 Dual-Core
GPU	ARM Mali400MP2 Complies with OpenGL ES 2.0/1.1
Memory (SDRAM)	1GB DDR3 (shared with GPU)
Onboard Storage	SD (Max. 64GB) / MMC card slot UP to 2T on SATA disk
Onboard Network	10/100/1000 Ethernet RJ45 (optional USB WIFI Dongle)
Camera Input	A CSI input connector allows for the connection of a designed camera module
Sound Input	Mic
Video Outputs	HDMI, CVBS , LVDS/RGB
Audio Output	3.5 mm Jack and HDMI
Power Source	5 volt via MicroUSB(DC In Only) and/or MicroUSB (OTG)
USB 2.0 Ports	2 (direct from Allwinner A20 chip)
Buttons	Reset button: Next to MicroUSB connector Power button: Next to Reset button UBoot button (optional): Behind HDMI connector
GPIO(2X13) pin	GPIO,UART,I2C bus,SPI bus with two chip selects,CAN bus,ADC,PWM,+3.3v,+5v,ground.
LED	Power Status LED (Red) Ethernet Status LED (Blue) User Define LED (Green)
Remote	IR

OS Support

- Raspbian
- Lubuntu
- OpenSuse
- Fedora
- ArchLinux
- Bananian
- Android 4.4

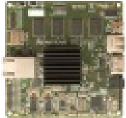
Category:

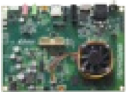
- Banana Pi

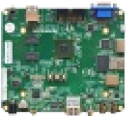
> From: eLinux.org

R-Car

Jump to R-Car boards pages ↴

 Start.Now (Stout)

 Porter

 SILK

Contents

Introduction

This is starting page for Embedded Linux on Renesas R-Car family of automotive embedded SoCs. Main goal is to make information on R-Car SoCs friendly, easy to access for embedded developers, enthusiasts. It does not aim to replace official Renesas support.

Links below could be used to get familiar with Renesas Automotive SoCs and R-Car Concortia:

- Renesas Automotive Products / Solutions
- Renesas Car Information Terminal Applications
- R-Car Consortium

Detailed information regarding R-Car SoCs (including roadmap, technical and marketing documents, BSPs and middleware) can be obtained directly from Renesas representative.

R-Car SoCs

Renesas R-Car SoC variants (Generations)

	Generation 1	Generation 2	Generation 3
"H" - Premium/High end	**H1** - R8A7779	**H2** - R8A7790	**H3** - R8A7795
"M" - Mid range	**M1A** - R8A7778	• **M2-W** - R8A7791 • **M2-N** - R8A7793	
"E" - Entry class	**E1**	**E2** - R8A7794	
"V" - ADAS		**V2H** - R8A7792	

R-Car Boards

Generation 1

Renesas R-Car Gen1 evaluation boards

	Standard	Low Cost Boards (LCB)
R-Car H1	Marzen	Geuze
R-Car M1A	BOCK-W	Milan
R-Car E1	Silverstone	

Generation 2

Renesas R-Car Gen2 evaluation boards

	Standard	Low Cost Boards (LCB)
R-Car H2	Lager	Start.Now (Stout)
R-Car M2-W	Koelsch	Porter
R-Car M2-N	Gose	
R-Car E2	Alt	SILK
R-Car V2H	Blanche	

R-Car Linux

Upstream

Renesas mainline R-Car Linux tree is maintained by Simon Horman

Start here to get latest/closest upstream kernel: git://git.kernel.org/pub/scm/linux/kernel/git/horms/renesas.git

```
e.g. pull development branch: git clone -b devel git://git.kernel.org/pub/scm/linux/kernel/git/horms/renesas.git
```

TBD - add page regarding R-Car upstream development process (including branches/tags meaning)

R-Car Community

- Mailing list - via kernel.org linux-sh Majordomo.

Archives available here: http://marc.info/?l=linux-sh , http://dir.gmane.org/gmane.linux.ports.sh.devel , http://www.spinics.net/lists/linux-sh/

LTSI (Long Term Support Initiative)

Renesas SoCs and boards are supported as part of LTSI project - http://ltsi.linuxfoundation.org/ . Refer to 3.10 and 3.14 baselines

Renesas Stable R-Car BSP Tree

Stable Renesas R-Car BSP Tree located here: git://git.kernel.org/pub/scm/linux/kernel/git/horms/renesas-backport.git. Pull latest stable branch.

```
for example: bsp/v3.10.31-ltsi/rcar-gen2-1.8.0
```

R-Car U-Boot

R-Car U-Boot mainline development is done using git://git.denx.de/u-boot-sh.git tree. It is maintained by Nobuhiro Iwamatsu.

```
e.g. pull renesas/bsp/rcar-gen2-1.8.0 branch and build U-Boot for Lager/Koelsch/Alt boards
git clone -b renesas/bsp/rcar-gen2-1.8.0  git://git.denx.de/u-boot-sh.git
```

build example

```
make O=/tmp/build clean
make O=/tmp/build mrproper
make O=/tmp/build gose_config
make O=/tmp/build all
```

R-Car Yocto

Renesas Yocto/Poky distribution is maintained by Nobuhiro Iwamatsu.

To get started with Yocto on R-Car Gen2 platform - pull git://git.yoctoproject.org/meta-renesas tree

If you pulled complete Yocto/poky trees, you should be able to kick-off a build using the 'oe-init-build-env' script in Poky. Update bblayers.conf by adding the meta-renesas and meta-rcar-gen2 layer. e.g.:

```
BBLAYERS ?= " \
        <path to layer>/poky/meta \
        <path to layer>/poky/meta-yocto \
        <path to layer>/poky/meta-yocto-bsp \
        <path to layer>/meta-renesas \
        <path to layer>/meta-renesas/meta-rcar-gen2 \
"
```

To build a specific target BSP configure the associated machine in local.conf:

```
MACHINE ?= "<supported board name>" (lager, koelsch, gose, alt)
```

Build the target file system image using bitbake:

```
$ bitbake core-image-minimal
```

R-Car Tizen

TBD *Renesas R-Car Tizen is already available via tizen.org trees. Add detailed instructions how to get started with Tizen for R-Car*

R-Car Android

TBD *This section will be added later*

> From: qualcomm.com

DragonBoard

> The DragonBoard™ platform is a development kit based on a Snapdragon processor. The kit exposes the Snapdragon processor and lets you take advantage of pins, connectors, adapters, and expansion headers to tap into its functionality in your own product development efforts. The DragonBoard development kit is ideal for creating and prototyping hardware components, developing middleware, specifying and testing embedded and IoT systems, conducting research, and teaching and learning electronic engineering.

- DragonBoard 410c

Embedded Linux Information

Resources or information about the embedded Linux ecosystem, from vendors, products, communities to jobs.

> From: eLinux.org

Products

Here is a list of products which use embedded Linux.

Please see the talk page for information about how we want to populate this page.

Contents

Televisions

Televisions using Embedded Linux

Product Name	Company Name	Year Introduced	RAM MB	Flash MB	Processor	Source or Notes	Download Area
X	X	X	X	X	X	X	X

Mobile Phones

Mobile Phones using Embedded Linux

Product Name	Company Name	Year Introduced	RAM MB	Flash MB	Processor	Source
VM1188T	Accton	2006	-	-	-	LinuxDe
C8000	Cellon	2005	-	-	-	LinuxDe
3G Linux Ref Design	Datang	2004	-	-	-	LinuxDe
E28 FMC phones	E28	2006	-	-	OMAP 730	LinuxDe
E28 E2800	E28	2003	32	32	ARM9	LinuxDe
E28 E2800+	E28	2004	64	32	ARM9 300MHz	LinuxDe
PWG500	G-Tek	2006	X	X	X	LinuxDe
G500i	Grundig	2005	52 - User	X	OMAP850	LinuxDe
N60	Haier	2006	X	X	X	LinuxDe
GPS Phone	ImCoSys	2006	64	64	OMAP 730	LinuxDe
Openmoko Freerunner	Openmoko	2008	128	256	armV4@400Mhz	Openmc open hardwar freely av
OpenPhoenux GTA04 / Letux 2804	Golden Delicious Computers	2011	512	512/1024	OMAP3 SoC (DM3730, ARMv7@800MHz/1GHz)	GTA04 v open ha (schema available
Nokia N900	Nokia	2009	256	256M NAND +32G eMMC	OMAP 3430 SoC(600 MHz ARM Cortex-A8 CPU)	N900 on
Htc Dream	HTC	?	192M(but some reserved for the DSPs)		MSM7201A(armv6-novfp@528Mhz)	run andr default,C port in progress routing c

Audio and Video Mobile Players

Audio and Video Mobile Players using Em

Product Name	Company Name	Year Introduced	RAM MB	Flash MB	Processor
PMA430	Archos	?	64MB(48MB usable)	?(loop image on the hdd used)	omap1(armv4@74.34 bogomips)
Archos 5	Archos	?	128 M	40G/60G/120G/160G/250G hdd	OMAP3
Archos 7	Archos	?	128 M	160G/320G hdd	OMAP3

Internet tablet

Internet tablet using Embedded Linux

Internet tablet using Embedded Linux

Product Name	Company Name	Year Introduced	RAM MB	Flash MB	Processor	Source or Notes	
N800	Nokia	?	128MB	256MB	OMAP2420@400 MHz		yes ne
WeTab	Neofonie/Pegatron	2010	1 GB	16 GB	1.66GHz Intel Atom N450	Running MeeGo	http:/ and-h

Settop Boxes

Settop Boxes using Embedded Linux

Product Name	Company Name	Year Introduced	RAM MB	Flash MB	Processor	Source or Notes	Download Ar
Freedom Jump	Amino Communications	2010	1 GB DDR 3	512 MB	Intel® Atom CE4150	MeeGo platform	http://www.am jump/
Roku SD/HD/HD-XR/XDS	Roku	2010	?	?	NXP PNX8935		Roku open so

Digital Video Recorders

Digital Video Recorders using Embedded Linux

Digital Video Recorders using Embedded Linux

Product Name	Company Name	Year Introduced	RAM MB	Flash MB	Processor	Source or Notes	Download Area
X	X	X	X	X	X	X	

Game Consoles

Game Consoles using Embedded Linux

Product Name	Company Name	Year Introduced	RAM MB	Flash MB	Processor	Source or Notes
PlayStation 2	Sony Computer Entertainment	2000	32MB	unknown	MIPS 64-bit R5900	PS2 wikipedia page, Playstation.linux.communi site
PlayStation 3	Sony Computer Entertainment	2006	512 MB	unknown	CELL broadband engine	PS3 wikipedia page
Didj	LeapFrog Enterprises	2008	32MB	256MB	LF-1000/Pollux	Didj
Leapster Explorer	LeapFrog Enterprises	2010	64MB	512MB	LF-1000/Pollux	Leapster Explorer
LeapPad Explorer	LeapFrog Enterprises	2011	64MB	2GB	LF-1000/Pollux	Leappad Explorer

Network Attached Storage (NAS)

Network Attached Storage using Embedded Linux

Product Name	Company Name	Year Introduced	RAM MB	Flash MB	Processor	Source or Notes	Download Area
NSLU2	Linksys	2004	32MB	8MB	Intel XScale IXP420		http://www.nslu2-linux.org (community site with improved firmwares)

Network Camera

Network Camera using Embedded Linux

Product Name	Company Name	Year Introduced	RAM MB	Flash MB	Processor	Source or Notes	Downloa
NC353L	Elphel, Inc	2007	64MB	128MB	Axis EtraxFS	http://wiki.elphel.com	http://sou
MayGion MIPS IPCam EyeSight ES-IP902W	MayGion	2011-04	27MB	4MB	Ralink RT5350 (MIPS 24K)	Firmware Provides telnet+FTP access to the Linux filesystem out of the box	*No sourc*

Networked Media Players

Networked Media Players using Embedded Linux

Product Name	Company Name	Year Introduced	RAM MB	Flash MB	Processor	Source or Notes	Download Area
MusicPal	Freecom	2007	32MB	unknown	Marvell 88W8618		http://www.musicpal (under Legal Notice

Still and Handheld Video Cameras

Still and Handheld Video Cameras using Embedded Linux

Product Name	Company Name	Year Introduced	RAM MB	Flash MB	Processor	Source or Notes	Download Area
HDR-UX1, HDR-SR1 video cameras	Sony	2006	??	??	ARM9	Sonystyle store listing	Sony Linux Download area

Assistive Technology

Assistive Technology devices using Embedded Linux

Product Name	Company Name	Year Introduced	RAM MB	Flash MB	Processor	Source	Download Area
Lightwriter SL38	Toby Churchill Ltd	2004	64	256	StrongARM SA-1110	Developer site	code and sources

Robots and Smart-companions

Robots and smart-companions using Embedded Linux

Product Name	Company Name	Year Introduced	RAM MB	Flash MB	Processor	Source or Notes	Download Area
Karotz	Mindscape then Aldebaran Robotics	2011	64	256	Samsung S3C2440 (ARM920T@400Mhz)	Developer site	

Network devices

Network devices using Embedded Linux							
Product Name	**Company Name**	**Year Introduced**	**RAM MB**	**Flash MB**	**Processor**	**Connectivity**	**Note**
Sheevaplug	Marvell		512M	512M	Kirkwood 88F6281 (armv5@1.2Ghz)	Ethernet,usb,sdio	

Modular devices

Modu					
Product Name	**Company Name**	**Year Introduced**	**RAM MB**	**Flash MB**	**Processor**
bugbase 1.2	Bug Labs	2006	128	32	imx31(armv6-vfp@533Mhz)
bugbase 1.3	Bug Labs	2009	128	32	imx31(armv6-vfp@533Mhz)
Overo fire	Gumstix	?	256M	256M	OMAP 3530(also contain POWERVR SGX + C64x)
overo air	Gumstix	?	256M	256M	OMAP 3503(no 3D acceleration and no DSP)
overo water	Gumstix	?	256M	256M	OMAP 3530(also contain POWERVR SGX + C64x)
Overo earth	Gumstix	?	256M	256M	OMAP 3503(no DSP,no 3d acceleration)
TNY-A9G20-LPW-I02	calao systems	2009	64M	256M	AT91SAM9G20(400Mhz)

TNY-A9263-C02	calao systems	2008	64M	256M	AT91SAM9263(200Mhz)
TNY-T3730-ULP-C01	calao systems	2011	256M	256M	OMAP3730(800Mhz)
QIL-A9260-C02	calao systems	2011	64M	256M	AT91SAM9260(200Mhz)
USB-A9G20-LPW-C02	calao systems	2011	64M	256M	AT91SAM9G20(400Mhz)
USB-A9263-C02	calao systems	2009	64M	256M	AT91SAM9263(200Mhz)
SKY-T3359-LPW-I01	CALAO Systems	2013	512M	512M	AM3359(720Mhz)
SKY-S200-IMX6-LPW-I01	CALAO Systems	2013	1GB	512M	iMX6 Quad Core (800MHz)
S200-A5D35-LPW-I01	CALAO Systems	2013	512M	256M	SAMA5D35(536Mhz)
Mizar32 A/B/C	SimpleMachines	2011	64K SRAM, 32M SDRAM/64K SRAM, 32M SDRAM/32K SRAM, 32M SDRAM	512K/256K/128K	AVR32 UC (66 MHz)

Development boards

Development boards

Development boards

Product Name	Company Name	Year Introduced	RAM MB	Flash MB	Processor	Conn
Beagle Board	Texas Instruments	2008	256M	256M	OMAP3530	
BeagleBone	Texas Instruments	2011	256MB	none (MicroSD)	AM3358	Etherne
Pandaboard	Texas Instruments	2010	8Gb(x32) x 2ch	(2G(x32) x4pcs) 2CS/ch	OMAP4430	WLAN, 10/100
Snowball SDK & PDK	CALAO Systems	2011	1GB	4GB / 8GB	ST-Ericsson A9500 Dual Cortex A9 + Mali 400	WiFi, Bl GPS, HI CVBS, E 10/100, Out, Mic Serial P console MiPi34, Acceler Magnet Gyrome Pressur 3x expa connect
SKY-T3359-LPW-I01	CALAO Systems	2013	512M	512M	AM3359(720Mhz)	Micro- Devic USB Ho Eth 10/100/ PCI E (USE ACC/M 3D, Hu Temp sensor, Sens Blue CANBu (Line In. Headph In), L expan
US$ 25 Raspberry Pi Computer	Raspberry Pi Foundation	2012	256	none (SD Card)	Broadcom ARM11 (ARM v6) + Videocore GPU	10/100 I HDMI , Compos Audio, U GPIO, J MIPI DS
Colibri Evaluation Board	Toradex	2012	up to 512MB	up to 1GB	Nvidia Tegra 2, Tegra 3, Marvell PXA270, 320, 310, 300	10/100 I HDMI , LCD, Au Host/ O GPIO, J Camera RS485, CAN, I2

						SDIO, M Bus, IrD
NanosG20	Ledato	?	64M/128M	128M/512M		
Carambola	8devices	2013	32 MB	8 MB Flash	MediaTek: Ralink RT3050 (320MHz MIPS24K)	USB, W Etherne RS232,
Carambola 2	8devices	2013	64 MB DDR2	16 MB Flash	AR9331 400 MHz	USB, W Etherne SPDIF, : GPIO, I2 antenna connect
Cubieboard	Cubietech	2012	512MB/1024MB DDR3 @480Hz	4Gb Nand Flash	Cortex-A8 + Mali400	2x USB, SATA, m HDMI, A IN/OUT, SPI, RG CSI/TS, ADC, C VGA, SI OUT, R-
VAB-600 Springboard	VIA	2013	1Gb DDR3	4Gb eMMC Flash	800MHz Cortex-A9 + Mali-400	10/100 I HDMI, C SPI, LVI Optiona Wi-Fi
VAB-800	VIA		1Gb DDR3	4Gb eMMC Flash	800MHz Freescale i.MX537 Cortex-A8 + AMD Z430	10/100M mini-HD USB 2.0 device, GPIO, L JTAG, A
VAB-820	VIA		1Gb DDR3	4Gb eMMC Flash	1GHz Freescale i.MX6Quad Cortex-A9 + Vivante GC2000	USB 2.0 USB2.0 COM pc wire DT COM/C/ FlexCAI I2C, SP Audio, n
VAB-1000	VIA		2Gb DDR3	4Gb eMMC Flash	1.0GHz VIA Elite E1000 Cortex-A9 dual-core + integrated graphics	1000M, HDMI, S LVDS, U COM, I2 GPIO, J miniPCI Optiona Audio
APC 8750	VIA	2012	512Mb DDR3	2Gb NAND Flash	800Mhz ARMv6 processor	10/100 I HDMI, V 2.0 (x4), out / Mic microSD
APC Rock	VIA	2012	512Mb DDR3	4Gb NAND Flash	800Mhz Cortex-A9 + Mali-400	10/100 I HDMI, V 2.0 (x4), microUS GPIO, S Audio ou microSD

Ethernut 5	egnite GmbH	2011	128 MB	1GB NAND Flash	AT91SAM9XE	USB 2.(Base-T 2xSPI, 2 1xI²S, in Sensor i 15 GPI(convert
US$99 Z-turn Board	MYIR	2015	1GB DDR3	16M QSPI Flash	Xilinx Zynq-7010/20	USB_U/ USB2.0 10/100/1 Etherne HDMI, T sensor, Temper Sensor
US$99 Rico Board	MYIR	2015	512MB DDR3	4MB eMMC Flash, 16M QSPI Flash	TI AM437x	UARTs, Host/De Gigabit Dual-Ca HDMI, L

Development boards

Product Name	Company Name	Processor	RAM MB	Flash MB	Video Display	Interfaces	
DevKit8000	Embest	TI OMAP3530	256MB DDR	512MB Nand Flash	LCD, VGA, DVI-D and S-Video	UART, USB Host, USB OTG, Ethernet, Audio, SD, Keyboard, JTAG	Linux
DevKit8500D/A	Embest	TI DM3730/AM3715	512MB DDR	512MB Nand Flash	LCD, VGA, DVI-D and S-Video	UART, USB Host, USB OTG, Ethernet, Audio, SD, Keyboard, JTAG	Linux Andr
DevKit8600	Embest	TI AM3359	512MB DDR3	512MB Nand Flash	LCD, VGA	UART, USB Host, USB OTG, Ethernet, CAN, RS485, WiFi/BT, Audio, TF, JTAG	Linux WinC
SABRE Lite	Embest	Freescale i.MX 6Quad	1GB DDR3	2MB SPI Flash	RGB, LVDS and HDMI	UART, USB, Ethernet, CAN, SATA, SD, JTAG, I2C, Audio	Linux Andr
		Atmel AVR32:		8M serial +	"LCD	UARTs, USB slave, 2x 100MBit	Linux

		AT32AP7000		8M parallel	Controller"	Ethernet, I2C, SPI, JTAG, AC97	http:/
MYD-AM335X	MYIR	TI AM3359	512MB DDR3	512MB Nand Flash	LCD, HDMI	UART, 4 x USB Host, USB OTG, 2 x Ethernet, CAN, RS485, Audio, TF	Linux WinC
MYD-SAMA5D3X	MYIR	Atmel SAMA5D3	512MB DDR2	256MB Nand Flash	LCD, HDMI	UART, USB Host, USB OTG, Ethernet, CAN, RS485, Audio, SD	Linux
MYD-IMX28X	MYIR	Freescale i.MX28	128MB DDR2	256MB Nand Flash, 128KB SPI Flash	LCD	UART, USB Host, USB OTG, 2 x Ethernet, 2 x CAN, RS485, Audio, SD	Linux
Virturilka	Virt2Real	TI DM365	128MB DDR2	256MB Nand Flash	Parallel camera interface, Analog component video output	GPIO, PWM, I2C, SPI, UART, DAC/ADC, USB, microSD, 10/100Mb RJ45 Ethernet	Linux

Development Boards with FPGA

Development Boards with FPGA

Product Name	Company Name	Year Introduced	RAM MB	Flash MB	Processor	Source	Download Area
APF27	Armadeus Systems	2009	up to 256	up to 512	i.MX27 400MHz + Spartan3a	Company Site	Developer Site

Development Boards with MINI2440v2 with 3.5

Development Boards with MINI2440V2

[1]

Development Boards with MINI2440V2

Product Name	Company Name	Year Introduced	RAM MB	Flash MB	Processor	Source	Download Area
[2]	2009	up to 256	up to 512	i.MX27 400MHz + Spartan3a	[3]	Developer Site	

Not Yet Categorized

Not Yet Categorized devices using Embedded Linux

Product Name	Company Name	Year Introduced	RAM MB	Flash MB	Processor	Source	Download Ar
Indamixx 2 Audio Tablet	Indamixx	2010	2 GB	150 GB HDD	Atom N450	Company website	http://www.ind 60.html
HS210	Ericsson	2000	32	32	StrongARM SA-1110	LinuxDevices	X
Armadillo-500	Atmark Techno	2007	64	16	i.MX31	Company Site	Armadillo Dev
Armadillo-300	Atmark Techno	2006	64	8	ARM9 200MHz	Company Site	Armadillo Dev
Armadillo-9	Atmark Techno	2004	64	8	ARM9 200MHz	Company Site	Armadillo Dev
Armadillo-240	Atmark Techno	2006	64	8	ARM9 200MHz	Company Site	Armadillo Dev
Armadillo-230	Atmark Techno	2006	32	8	ARM9 200MHz	Company Site	Armadillo Dev
Armadillo-220	Atmark Techno	2006	32	8	ARM9 200MHz	Company Site	Armadillo Dev
Armadillo-210	Atmark Techno	2005	32	4	ARM9 200MHz	Company Site	Armadillo Dev
Eddy v2.1 Series	SystemBase	2008	32	8	ARM9 210MHz	Company Site	Eddy Develop
Matrix Series	Artila	2006	64	16	ARM9 180MHz	Company Site	Matrix Site

Home And Building Automation, Smart Energy

HABA & Smart Energy controllers using Embedded Linux

Product Name	Company Name	Year Introduced	RAM MB	Flash MB	Interfaces	Processor	Source
Duckbill	I2SE GmbH	2013	128	-	UART, SPI, I2C, Ethernet, USB 2.0 Device, ADC, GPIO, micro-sd	i.MX283	Downloads
HABA-KNX-EXPLORER-C01	CALAO Systems	2011	128	256		AT91SAM9G20 (400MHz)	Repository
HABA-KNX-LITE-E01	CALAO Systems	2012	128	256		AT91SAM9G20 (400MHz)	Repository

Category:

- Products

Companies

Contents

Overview

This page provides information about companies that build and sell consumer electronics devices with Linux as their operating system. If you're looking for companies that build and sell Linux distributions for embedded devices or who provide services around embedded Linux, please see the Vendors page.

You may also want to look at the Source code download sites page, which has a list of places to get open source software from different companies.

A

- Acme Systems srl
    - Aria G25 SoM
    - FOX Board G20
    - Terra Board
- Aldebaran Robotics
- Archos
- Atmark Techno

- Armadillo Series
- SUZAKU Series
- ARMadeus Systems
  - APF27 Series (i.MX27 + Spartan3A based boards)
  - APF9328 Series (i.MXL + Spartan3 based boards)
  - The Armadeus Project: non profit association for embedded Linux geeks
- Artila Electronics
  - Matrix series
  - System on Module
  - PAC series

B

- Bluewater Systems
  - ARM based modules running Linux
- Beyond Semiconductor an obscure Slovenian company licensing processor cores using own BA1 and BA2 instruction set. If you happen to find something like Beyond BA14 or Beyond BA25 in some random device you know where it came from.

C

- CALAO Systems
  - Embedded Computers Expansion Boards Development Boards
  - USB Keys Home And Building Automation Controllers
- CompuLab Ltd.
  - ARM based modules running Linux
  - NetTop running Linux
- Crank Software
  - Storyboard Suite Embedded User Interface
  - Embedded Consulting Services
  - Embedded WebKit Development
- Crystalfontz America, Inc.
  - CFA-10036 i.MX28 based SOM
  - CFA921-TS CFA-10036+integrated display with touch
  - CFA920-TS CFA-10036+integrated display with touch
- CubieTech
  - cubieboard
- CWLinux
  - Single-board SysOnDIMM computers
  - OEM customized platforms

D

- DataPatterns India Pvt Ltd
  - Powerpc Based Single Board Computers

E

- Embest provides standard single board computers and custom modules based on different ARM processors from

Atmel, Freescale, NXP, Samsung, STMicroelectronics and Texas Instruments for embedded applications
- Electronic Engineering Solutions
- egnite GmbH produces the development boards of the Ethernut project.
- Elphel, Inc provide high performance Network Cameras based on Free Software and Hardware designs. Axis EtraxFS & Spartan 3e 1200k gates FPGA.
- einfochips - The Solution People

G

- Garmin
  - Source code for Linux-based products (Nuvi 8xx and 5xxx series)
- GlobalScale Technologies hardware and software design
  - SheevaPlug and GuruPlug

H

- hardkernel - Korean company producing
  - ODROID line of development boards
- HITEG LTD - Company focus on embedded board,single board computer
  - Developmentboard

I

- I2SE GmbH
- iCube obscure Chinese company
  - IC1 processor with own MVP instruction set - supposedly an evaluation board is available but no known product based on this processor. Due to instruction set design GPU is not required for graphics and media processing.
- iEndian - Company formed to fund and manage production of
  - Balloonboard
- Intel - Wikipedia entry
  - CE2110 Media Processor
- Intellimetrix - Computing for Science and Industry
  - Embedded Linux Learning Kit

K

- kernel concepts
  - Linux embedded services, porting and support
  - Open Source projects and news
- KOAN sas
  - KaeilOS industrial grade embedded linux
  - Linux embedded support
  - Device drivers development

L

- LanMusic: Linux based Internet radio player for Hotels and home users
- Lemote Chinese company selling various products based on the Loongson processor.
- Logic Supply and visit inspire.logicsupply.com for BeagleBone Black tutorials and resources.

M

- moblin.org - Home of Intel's "Mobile Linux" distribution and tools
- Motorola - Wikipedia entry
- MontaVista - Wikipedia entry
- Mistral Solutions
- MYIR Tech Limited
  - ARM Development Boards, Single Board Computers, CPU Modules
    - MYD-AM335X (TI AM335x ARM Cortex-A8)
    - MYD-SAMA5D3X (Atmel ATSAMA5D3 ARM Cortex-A5)
    - MYD-IMX28X (Freescale i.MX28 ARM926EJ-S)
    - MYD-SAM9X5 (Atmel AT91SAM9G15/G25/G35/X25/X35 ARM926EJ-S)
    - MYD-SAM9X5-V2 (Atmel AT91SAM9G15/G25/G35/X25/X35 ARM926EJ-S)
    - Rico Board (TI AM437x ARM Cortex-A9)
    - Z-turn Board (Xilinx Zynq-7010/20 ARM Cortex-A9+FPGA)
  - Custom Design Services based on ARM processors

N

- Neuros Technology - Wikipedia entry
  - Neuros OSD
- Nokia - Wikipedia entry
  - N800
  - N770
- NVIDIA Corporation - Wikipedia entry
  - Tegra
- NXP Semiconductors - Wikipedia entry

O

- Hardkernel - Wikipedia entry

P

- Parkhelp: System that helps users locate parking spaces effectively using a system of Linux based embedded guided information
- Pengutronix
  - Linux Kernel Development GUI Development, Qt PTXdist Build System
  - Barebox Bootloader
  - i.MX Linux Port
- Pure
  - Evoke Flow
- ProFUSION embedded systems

- Guarana and Enjoy

R

- RidgeRun
- Route 495 Software

S

- Samsung - Wikipedia entry
  - Samsung Open Source Release Center
- Sony - Wikipedia entry
  - Global Linux source code download site
  - US Linux source code download site
- Secure Computing - Wikipedia entry
- SnapGear family of products
- Simtec Electronics - Hardware and Software design

T

- Timll A Chinese company focused on providing standard single board computer and custom modules for ARM embedded applications
- TechNexion
- TimeSys
  - LinuxLink
- TechnologicSystems at http://embeddedarm.com
- Tk Open Systems BSP's-R-Us, also drivers, even some user-space stuff
- TomTom Wikipedia entry
- Toradex Embedded Computer Modules
- Tvblob
  - vMAX vTALK vLINK Tvblob BOX

V

- VIA Embedded is a division of VIA Technologies, Inc., developing embedded x86 and ARM boards and systems, and providing hardware and software customization services.

W

- Wacom - is a world-wide company that produces graphics tablets and related products. Wacom's OEM sensor technology has been used by many major Tablet PC vendors. A detailed list of Wacom products and their histories can also be found at WikiWacom.
- Wilibox - specializes on embedded Linux based software platform and Wi-Fi stack for common access point and router hardware mostly for large scale networks. Productst and technologies can be found at: http://www.wilibox.com/products/software-and-hardware

X

- Xilinx - all programmable™
    - Zynq - All programmable SoC

0-9

- 8devices:
    - Carambola

Instructions for submitters

Please add company names in alphabetical order, and follow the existing format. Make a link from your company name to your main web site. If you have a wikipedia entry, please link that as well. If you would like to, you may list a few of your Linux-based products, but please don't go overboard.

Category:

- Companies

> From: eLinux.org

Vendors

Overview

This page lists companies who offer services around embedded Linux, such as training, support, etc. We also list companies who provide Linux distributions specifically targeting embedded uses. If you are looking for companies who build and sell devices with Linux as their operating system, please see the Companies page.

- A2e Technologies, USA (San Diego & Boston)
- Adeneo Embedded, France
- Armadeus Systems, France
- Bluewater Systems, New Zealand
- BEC Systems, USA
- CALAO Systems, France
- Codefidence Ltd., world wide
- Codethink Ltd., world wide
- Cogent Embedded, USA, Europe
- CWLinux, Hong Kong
- DENX Software Engineering
- Driving Devices Limited, United Kingdom
- e-con Systems Inc., world wide
- Embedded Alley, world wide (gone - was acquired by Mentor Graphics)
- Embedded Bits, United Kingdom, worldwide
- Embest, China (acquired by Premier Farnell/element14)
- Embedded Software Foundry Limited, Sheffield, UK (gone - acquired by William Matthew Limited)
- Essensium/Mind, Belgium
- FemtoLinux, Israel
- Free-Electrons, France
- Forlinx Embedded, China
- Holoscópio Tecnologia Ltda., Brazil
- Intellimetrix
- KAT Digital Corp.
- KOAN sas, Bergamo, Italy
- Lineo Solutions, Japan
- Linkodas, Lithuania
- Linutronix, Germany
- Mentor Graphics (embedded software group), worldwide
- MontaVista, world wide
- MPC Data, UK
- MYIR, China
- Openedev, India, Worldwide
- Opersys, North America
- Pengutronix, Germany
- Practical Control, UK
- RidgeRun
- Secret Lab, North America
- Sidebranch, The Netherlands.
- Signetik, North America
- Simantinee Embedded System, India
- Spore Tools, New Zealand

- SYSGO, Germany
- Team Embedded, Netherlands
- Technosolve Systems, India
- TimeSys, USA
- Tianyeit, China
- Timll, China
- WhiteQueue Consulting, New Zealand
- William Matthew Limited, United Kingdom
- Wind River, world wide
- Zeta2 GmbH, Switzerland
- ZhiYuan, Guangzhou, China

Category:

- Companies

> From: eLinux.org

Processors

Here is a list of different processor families, with miscellaneous notes for development information:

See also Hardware Hacking for a list of systems that include these processors.

Contents

ARC

The ARC 770D 32-bit RISC processor from the DesignWare ARC Processor IP family is optimized for running Linux. The latest version of ARC Linux is available from kernel.org and older versions as well as the GNU toolchain and several other packages like U-boot and Buildroot for ARC are available via GitHub.

More information from Synopsys:

- ARC Linux
- GNU Toolchain for ARC processors

ARM

See ARM website and the Wikipedia ARM article for information about the ARM architecture and processor family.

From the Linux perspective, there are 2 very different kinds of ARM chips:

- ARM processors that include a memory management unit (MMU), and can run standard Linux
- ARM processors without MMU. These can run a modified version of Linux called uClinux (http://uclinux.org/), enabling Linux to run on MMUless platforms or embedded processors with memory protection unit (MPU). These include ARM processors such as ARM7TDMI, ARM1156T2(F)-S or ARM Cortex-R4(F) for instance.

Please note that because of security considerations for MMU-less processors, it is unwise to use them when 3rd-party or untrusted code will be running on the device. For locked-down, single function devices, MMU-less processors may be appropriate. They are usually less expensive than processors with MMU.

Some major ARM platforms/SOCs are:

- Sitara, DaVinci and OMAP from Texas Instruments
    - TI lists where to find the Linux kernel GIT repositories for the broad market devices on their applications processors cross-reference
    - There is also an OMAP Linux kernel project
- i.MX - by FreeScale
    - Freescale's GIT repository for i.MX Linux support is at: http://opensource.freescale.com
        - Info about this repository, as of April 2007 is at: http://www.spinics.net/lists/arm-kernel/msg39771.html
- ARM RealView platforms - by ARM Ltd.
    - Linux BSP and resources available at http://www.arm.com/linux with associated GIT tree
- XScale/PXA - by Marvell (formerly Intel) -- has MMU
    - PXA255/PXA26x - Cotulla/Dalhart
    - PXA27x - Bulverde
    - PXA3xx - Monahans family
        - Linux PXA255/PXA26x/PXA27x BSPs are available in mainline kernel. You can find PXA3xx BSP from Marvell. Marvell team is working hard to get PXA3xx patches accepted by the mainline.
- Orion - by Marvell
    - Linux BSP for Orion-2 SoC available on ARM Linux Mailing List.
- Philips LPC21xx series of ARM processors are currently the lowest-cost ARM processors available. But they have no MMU.
- JuiceBox uses a ARM S3C44B0X. It runs uClinux.
- AT91 - by Atmel
    - AT91RM9200 - ARM920T based -- has MMU
    - AT91SAM9 Series
        - ARM926EJ-S based -- has MMU
    - SAMA5D3 Series
        - Cortex-A5 based -- has MMU
    - SAMA5D4 Series
        - Cortex-A5 based -- has MMU
    - Linux gateway : www.linux4sam.org
    - Github for Linux-based systems : linux4sam.org github
- Cirrus Logic (Linux forum and download site)
    - EP73xx - ARM720T based
    - EP93xx - ARM920T based
- Samsung Electronics (System LSI)
    - S3C24xx - ARM 9 Series
        - S3C2410 [1]
            - ARM920T
        - S3C2440 [2]
            - ARM920T
        - S3C2443 [3]
            - ARM920T
        - S3C2412 [4]
            - ARM926EJ
        - S3C2413 [5]
            - ARM926EJ
        - S3C2416 [6]
            - ARM926EJ
        - S3C2450 [7]
            - ARM926EJ
    - ARM 11 Series
        - S3C6400 - ARM1176

        - S3C6410
            - ARM1176, 800Mhz, 667Mhz, 533Mhz
        - S5P6440
            - ARM1176, 667Mhz, 533Mhz
        - S5P6442
        - S5P6450
    - Cortex-A8 Series
        - S5PC100
            - 833Mhz, 667Mhz
        - EXYNOS3110
            - 1Ghz
    - Cortex-A9 Series
        - EXYNOS4210
            - 1.2Ghz, 1Ghz
- Allwinner Technology (AllwinnerTech(Chinese))
    - A1x - Cortex-A8, 1~1.5Ghz
- Qualcomm Inc. [8]
    - Snapdragon S3 (Scorpion) "Snapdragon S3 (Scorpion)")
    - Snapdragon S4 (Krait) "Snapdragon S4 (Krait)")

MIPS

Information about MIPS processor architecture can be found here. For the Linux port information can be found here.

Processors based on MIPS architecture include

1. TX System RISC from Toshiba.
2. MSP series of processor from PMC Sierra.

SuperH



Built by Renesas Technology the webpage of record for the SuperH family of microprocessors can be found here: SuperH RISC Engine Family.

Wikipedia Page: SuperH

Linux on SuperH: linux-sh

Renesas SuperH Overview

SuperH is an embedded RISC developed for high cost-performance, miniaturization, and performance per unit of power consumption (MIPS/W). We are developing CPU cores for a wide range of applications and functions and have many products available. Our product lines include a series with the SH-2 as the CPU core and on-chip large-capacity flash memory and peripheral functions such as timer, serial I/O, and AD converter, and a series with the SH-3 or SH-4 as the CPU core, which achieves high-speed data processing and is equipped with cache and MMU. Additionally, there is lineup of series with the SH2-DSP or SH3-DSP as the CPU core, which have full DSP functions and an emphasis on multimedia and communications processing. Currently available products also have lots of features, such as low power modes, low power consumption, and small size. Various versatile operating systems and development tools have been improved, allowing for more efficient development.

Devices

- Sega
  - Dreamcast - Limited to the machine models that can start by MIL-CD and usage of a Broad Band Adapter is recommended.
- Hitachi ULSI Systems
  - MS7206SE01 - SH72060 Solution Engine
  - MS7750SE01 - SH7750(sh4) Solution Engine
  - MS7709SE01 - SH7709(sh3) Solution Engine
- SuperH, Inc.
  - ["MicroDev"]
- HP Jornada
  - 525 (SH7709 (sh3))
  - 548 (SH7709A (sh3))
  - 620LX (SH7709 (sh3))
  - 660LX (SH7709 (sh3))
  - 680 (SH7709A (sh3))
  - 690 (SH7709A (sh3))
- Renesas Technology Corp.
  - RTS7751R2D - CE Linux Forum（CELF）Compliant Evaluation Board
- Renesas Europe/MPC Data Limited
  - EDOSK7705 - SH7705 sh3
- EDOSK7760 - SH7760 sh4
  - EDOSK7751R - SH7751R sh4
  - SH7751R SystemH - SH7751R sh
- CQ Publishing Co.，Ltd.
  - CQ RISC Evaluation Kit(CqREEK)/SH4-PCI with Linux
  - [http://www.kmckk.co.jp/eng/ Kyoto Microcomputer Co., Ltd. (KMC or KμC)
  - Solution Platform KZP-01 KZP-01[Mainboard] + KZ-SH4RPCI-01[SH4 CPU Board]
- Silicon Linux Co,. Ltd.
  - CAT760 - SH7760
  - CAT709 - SH7709S
  - CAT68701 - SH7708R For A-one CATBUS[Designed for 68000 board] compliant
- Daisen Electronic Industrial Co., Ltd.
  - SH2000 - SH7709A 118MHz
  - SH2002 - SH7709S 200MHz
  - SH-500 - SH7709S 118MHz
  - SH-1000 - SH7709S 133MHz
  - SH-2004 - SH7750R 240MHz
- [IO-DATA DEVICE, Inc.(Network Attached Storage NAS Series)]
  - LAN-iCN - NAS Adapter for IODATA HDD with "i-connect" Interface
  - LAN-iCN2"] - NAS Adapter for IODATA HDD with "i-connect" Interface
  - LANDISK"] - SH4-266MHz[FSB133MHz] RAM64MB UDMA133 USB x2 10/100Base-T
    - HDL-xxxU - LANDISK Series NAS Standard Model
    - HDL-xxxUR - LANDISK with RICOH IPSiO G series print monitor for Windows support
    - HDL-WxxxU - LANDISK with wide body & twin drive support for Heavy storage or RAID1
    - HDL-AV250 - LANDISK with Home Network DLNA guideline support
    - LANTank - LANDISK kit SuperTank(CHALLENGER) Series
      - HDL-WxxxU based twin drive bulk NAS kit. LANTank have a special feature that supported network media server(cf. iTunes etc..).
- TOWA MECCS CORPORATION
  - TMM1000 - SH7709
  - TMM1100 - (SH7727

  - TMM1200 - SH7727
- Sophia Systems
  - Sophia SH7709A Evaluation Board
  - Sophia SH7750 Evaluation Board
  - Sophia SH7751 Evaluation Board
- MovingEye Inc.
  - A3pci7003 - Using SH7750/ART-Linux [Linux with Realtime Extension]
- AlphaProject Co., Ltd.
  - MS104-SH4 - SH7750R/PC104(Embedded ISA Bus) with apLinux
- Interface Corporation.
  - MPC-SH02 - SH7750S: ATX Motherboard Style
  - PCI-SH02xx"] - SH7750S: PCI-CARD Style
- TAC Inc.
  - T-SH7706LAN another name "Mitsuiwa SH3 board" SH-MIN - SH7706A/128MHz Flash512KB SDRAM 8MB 10BASE-T
- SecureComputing/SnapGear (older products, check ebay etc, all can netboot and have a debug header)
  - SG530 - SH7751@166MHz RAM16MB FLASH4MB 2x10/100 1xSerial
  - SG550 - SH7751@166MHz RAM16MB FLASH8MB 2x10/100 1xSerial
  - SG570 - SH7751R@240MHz RAM16MB FLASH8MB 3x10/100 1xSerial
  - SG575 - SH7751R@240MHz RAM64MB FLASH16MB 3x10/100 1xSerial
  - SG630 - SH7751@166MHz PCI NIC card RAM16MB FLASH4MB 1x10/100 1xSerial-header
  - SG635 - SH7751R@240MHz PCI NIC card RAM16MB FLASH16MB 1x10/100 1xSerial-header

PowerPC

Some processors and tools for PowerPC are

Processors

- Freescale MPC5200 SOC
- RAD750

Tools

The DENX Embedded Linux Development Kit (ELDK) provides a complete and powerful software development environment for embedded and real-time systems. It is available for ARM, PowerPC and MIPS processors and consists of:

- Cross Development Tools (Compiler, Assembler, Linker etc.) to develop software for the target system.
- Native Tools (Shell, commands and libraries) which provide a standard Linux development environment that runs on the target system.
- U-Boot - Firmware that can be easily ported to new boards and processors.
- Linux - Kernel including the complete source-code with all device drivers, board-support functions etc.
- Xenomai - Real-Time and RTOS Emulation Framework
- SELF (Simple Embedded Linux Framework) as fundament to build your embedded systems on.

All components of the ELDK are available for free with complete source code under GPL and other Free Software Licenses. Also, detailed instructions to rebuild all the tools and packages from scratch are included.

The ELDK can be downloaded for free from several mirror sites or ordered on CD-ROM for a nominal charge (99 Euro). To order the CD please contact office@denx.de

Detailed information about the ELDK is available here.

RT Patches

- See RTPatch For PowerPC

Documents

- Book E and PPC 440 - Descriptions about diffrences Book E and Ppc440.
- Ppc Wait Mode - PPC wait mode and sleep mode information

XScale

CE2110 Media Processor

- CE2110 Media Processor

The highly integrated Intel CE 2110 Media Processor helps to simplify the design of consumer electronics products with reduced BOM cost. The integrated Intel XScale® processor core at 1GHz provides processing performance and headroom to deploy new revenue-generating applications. Hardware-based decode of widely used video codecs (MPEG-2, H.264) maximizes system-level performance by enabling the processor core to be used exclusively for applications.

The Intel CE 2110 Media Processor also includes an Intel® Micro Signal Architecture (Intel® MSA) DSP core for audio codecs, a PowerVR* 2D/3D graphics accelerator, hardware accelerators for encryption and decryption, comprehensive peripheral interfaces, analog and digital input/outputs, and a transport interface for ATSC/DVB input.

- The Intel CE 2110 Media Processor Development Platform is designed to reduce time-to-market for new applications.
- The Intel CE 2110 Media Processor reference platform provides the foundation for rapid development of new customer designs and product demonstrations.

x86

- Geode from AMD
- AMD Geode GX / CS5535
- AMD Geode LX / CS5536
- Graphical representation of Intel x86 processors from i386 - present on Meld.org

AVR32

- AP7000 from Atmel

In 2010 Atmel stopped the further development of the AP7000 processor.

Blackfin

- Blackfin

m68k

The Freescale m68k family includes:

- 68xxx
- Coldfire
- DragonBall

Resources:

- Coldfire Mailing List

Further reading

- Several processors have their own wiki, listed on the WikiNode.

Categories:

- NeedsEditing
- Processors

> From: eLinux.org

Community

This page is for information about the embedded Linux and open source community.

Contents

Netiquette

Please read Netiquette before interacting with the Open Source Communities

- Patch_Submission_HOWTO

Community sites

General Portals

- Linux.com - Linux community portal sponsored by the Linux Foundation
- LinuxGizmos.com - the canonical (no pun intended) place for news about embedded Linux (and the successor to the now-inactive LinuxDevices.com, which is now available as a static, searchable 15,000 page archive)
- Linux.org - an excellent starting place for all things linux

Hardware-Specific Communities

- ARM Linux - the central place for Linux on ARM, this is where you find Russell Kings patch tracker for example
- Beagle Board community - Portal for the Beagle Board community, sponsored by TI
- OpenSourceMID.org community - Portal for the K7 OMAP3530 MID community
- PandaBoard community - Portal for the PandaBoard community, sponsored by TI
- Parallella community - Portal for the Parallella community, a $99 Linux supercomputer.

- Raspberry Pi community - Portal for the Raspberry Pi community, an ARM GNU/Linux box for $25. Take a byte!
- Snowball community - Portal for the Snowball community
- UDOO community - Portal for the UDOO community. Android, Linux and Arduino in a tiny single-board computer
- Improv community - Portal for the Improv community. A Modular ARM GNU/Linux and Android single-board computer system for everyone.
- ODROID community - Portal for the ODROID community
- NVidia Tegra community - Portal for the NVidia Tegra (e.g. Jetson TK1) community
- Sunxi Linux Community - Wiki for Allwinner SoC based devices - eg. some Olimex boards, Cubieboard, Banana pi, ...
- Exynos Linux Wiki - Wiki for Samsung Exynos based devices - eg, Odroid, some Chromebooks, ...

Software-Specific Communities

- Moblin community - portal for the Moblin community (merged with Maemo to form Meego - see next item)
- Meego community - portal for the Meego community
- Yocto Project - portal for the Yocto project

Communities for beginners

- Kernel newbies - General site for people getting started developing on the Linux kernel
- Japanese site for kernel newbies
- Embedded Systems Common Technical Baseline - Although not directly related to Linux this site is an excellent overview of what embedded systems are seen from various angles (hardware, software, design methods, etc...)
- Community Participation Guides
  - Resources for how to participate in Open Source Communities.

People

This section lists inviduals who are "movers and shakers" in embedded Linux: For more Information be sure to checkout MAINTAINERS

Linux kernel

Important kernel figures

- Linus Torvalds - Linux kernel initiator and head maintainer
- Andrew Morton - Maintains an important secondary (staging) tree
- David Woodhouse - Embedded Linux maintainer
- Matt Mackall - Embedded Linux maintainer, originator of Linux-tiny patch set (author of SLOB allocator), author of kpagemap and smem
- Greg Kroah-Hartman - Initiator and maintainer of the Linux Driver Project / Staging Tree and quite a handyman
- Stephen Rothwell - Maintainer of the Linux-Next-Tree, most stuff goes in there before getting merged into Linus' tree
- Paul Gortmaker - Embedded Linux maintainer
- Ted Tso - EXT4 maintainer, Kernel Summit organizer
- James Bottomley - SCSI maintainer, Linux Foundation technical advisory board chair

Kernel arch maintainers

- Arnd Bergman, Russell King - ARM kernel maintainers
- Ingo Molnar, Thomas Gleixner, Peter Anvin - x86 maintainers
- Paul Mundt - SH kernel maintainer
- Ralf Baechle - MIPS kernel maintainer
- Greg Ungerer - uCLinux/Coldfire kernel maintainer
- Haavard Skinnemoen - avr32 kernel Maintainer (Atmel)

Feature developers/maintainers

- David Woodhouse - MTD/jffs2 author, embedded Linux kernel maintainer
- Andi Kleen - Author of bloat-o-meter
- Ingo Molnar - Author of RT-preempt patch set, kernel scheduler maintainer, x86 maintainer
- Phillip Lougher - Author of Squash FS
- Jason Wessel - KDB maintainer
- John Stultz - Mainliner of many Android technologies
- Thomas Gleixner - RT-preempt maintainer, x86 maintainer, IRQ subsystem maintainer
- Colin Cross - Google Android developer

Other People

- Jason Wessel - Wind River Linux Architect
- Sean Hudson - Mentor Graphics Embedded Linux Architect
- David Rusling - CTO of Linaro
- Richard Purdie - Poky originator, Yocto Project architect
- David Stewart - Yocto Project leader
- Bradley Kuhn - SFLC license enforcement agent
- David Anders - TI contractor, elinux wiki contributor
- Tim Bird - Sony Linux researcher, CE WG AG chair, ELC organizer
- Paul Walmsley - OMAP kernel developer
- Ben Dooks - ARM/Samsung arch kernel maintainer
- Catalin Marinas - ARM kernel developer (developer of numerous ARM BSPs, as well as kmemleak author)
- Thomas Petazzoni - Marvel SOC kernel developer
- Michael Opdenaker - Free Electrons founder

Interview candidates

The following page has a list of people we'd like to interview for an eLinux.org feature:

- Interviews

Foundations and Forums

- ARM Development Discussion Forum launched by Embest
- Embest Product User Forum
- CE Workgroup of the Linux Foundation (formerly the CE Linux Forum)
  - Embedded Linux Conference Presentations
  - CE Workgroup page on the Linux Foundation site
  - (deprecated: old CELF home page)
- Linux Foundation

Linux User Groups

One way to get involved with a bunch of like-minded Linux enthusiasts is to participate in a local Linux users group. The following site has a good database of Linux users groups:

- www.linux.org/groups
- CLUE LUG List - Canadian User Group listing.
- Bangalore Beagle User Group Meet

Development Model

- The Cathedral and the Bazaar
- FIXTHIS - add more links to papers and articles about the development model

Reasons for contributing to open source

- Open Source ROI Model - a page about return on investment from open source contributions

Community-building ideas

See a discussion thread on this at: http://lists.celinuxforum.org/pipermail/celinux-dev/2012-September/000637.html

In 2012, Tim Bird proposed 4 activities to encourage more embedded Linux collaboration and community-building:

1. Resurrect the celinux-dev mailing list
2. Ideas:
    - Use it more often for discussion
    - Promote on other lists (linux-embedded...?)
    - Introduce people who are using it to each other
3. more focused groups at upcoming conferences.
4. Ideas:
    - have informal meeting places for specific topical areas
5. I'm looking for ways to invigorate the elinux.org wiki site.
6. Ideas:
    - more contests?
7. more involvement in and use of the LTSI kernel
8. Ideas:
    - ???

Other ideas:

- a 'planet' feed or news feed for embedded linux users
    - possibly: planet.elinux.org (Bill Traynor is working on it)
    - Thomas Petazzoni made one here: http://www.emlinews.net/
        - go to http://www.emlinews.net/submit/ to submit a news article
- help bridge between embedded and non-embedded mainline developers (a'la Arnd Bergmann)
    - see reference to Arnd here: http://lists.celinuxforum.org/pipermail/celinux-dev/2012-September/000645.html
- Better/open discussion on funded projects for embedded Linux
    - possible projects: mini-distro for software update
    - create a Google hangout for a discussion on this

Quality Assurance

This section has links to aspects of the development model designed to provide quality assurance.

Certificate of Origin

Developers who contribute code to the Linux kernel agree to the Developer Certificate Of Origin by signing their code, with a "Signed Off By" line.

Categories:

- NeedsEditing
- Community

> From: eLinux.org

Experts

Contents

Embedded Linux Expert List

- Rusty_Russell_Quotes This page is here in honor of Rusty Russell, one of the funniest Linux kernel developers EVER

Instructions for Experts

This page is intended to provide a place for Embedded Linux Experts to advertise their availability, and describe their expertise, to companies interested in paying them for services. This could include contract work or fulltime employment.

Linux experts are encouraged to provide their information in the table below so that companies may contact them for their areas of interest. Please put an expiration date for the information, so we can remove information when it gets stale. Or even better, when you have obtained work, please remove your entry from this table.

See the Jobs page for jobs already posted to this site.

Instructions for Employers

Please contact the person below, if you have a position or contract work you would like to hire someone to do. This list is not intended to be used to ask for free support.

You can also list your job on the Jobs page.

The List

Name (Link)	Expertise areas	Contact Info	Expires date
Andrew Murray Embedded Bits Limited UK	Embedded Linux: Boot Time Reduction, Board ports, Driver development, OMAP4.		2016
Constantine Shulyupin Israel, Worldwide	TI DaVinci, drivers, boot loaders, fast boot	http://www.makelinux.com	2015
Leon Woestenberg, MSc. EE, Eindhoven, The Netherlands, EU	Embedded Linux, Real-time, OpenEmbedded, System and Hardware Design, Device Drivers, Training, Video Broadcast, FPGA's, PCI Express	www.sidebranch.com	2015

Ithamar Adema, Mark Vels, Netherlands, EU	Embedded Linux, BSP ports & support, OpenWrt support, Device Drivers, Training,	www.team-embedded.com	2013
Robert Berger - Reliable Embedded Systems, Embedded Software Specialist, Athens (Greece)/Mitterdorf (Austria), EU	Consulting and Training for: Embedded Linux (Systems Architecture, Device Drivers), Real-time, Debugging, U-boot, ...	elinux@reliableembeddedsystems.com http://www.reliableembeddedsystems.com	2043
Marco Cavallini, Bergamo (Italy), EU	Embedded Linux, Device Drivers, Real-time, OpenEmbedded, BSP Design, u-boot, training	KOAN sas	2019
Young-Ho Song, Seoul (Korea)	Embedded Linux, Device drivers, Multimedia Communication & System, IPTV, Network Security	songyounghoATkorea.kr whois Young-Ho Song	2017
Frank Duason, Massachusetts (USA)	Embedded Linux, board bring-up, u-boot, BSP, device drivers, PowerPC, ARM, power management, applications and libraries	fduason@yahoo.com	2013
Thomas Soehus, Aalborg (Denmark)	TI OMAP, Sitara, Davinci and Freescale iMX SOC's, Embedded Linux, U-boot, BSP, device drivers, Yocto and Android.		2018
Juan Solano, Munich (Germany)	Embedded Linux, ARM, Device Drivers, Real-time.		2015
Steve Poulsen - Signetik, Nebraska (USA)	Embedded Linux, ARM/DSP, System Bring-up including HW troubleshooting, DSPLink, DSPBios, Device Drivers, Real-time.		2018
Aaron Clarke (USA) Embedded Consultant e-Linux User Page	Prototyping; BeagleBone, Raspberry Pi board bring-up; Linux kernels, bootloaders and device drivers peripherals and communications protocols troubleshooting and integration	web: AaronClarke.com e-mail:	2016

Category:

- Categories

> From: eLinux.org

Jobs

Contents

Embedded Linux Jobs List

Instructions for Employers

This page is intended to provide a place for people or companies to advertise their embedded Linux jobs or contract work. This could include contract work or fulltime employment.

Companies with Linux work are encouraged to provide their information in the table below so that Linux experts may contact them for jobs of interest. Please put an expiration date for the information, so we can remove information when it gets stale. Or even better, when the work is no longer available, please remove your entry from this table.

You can also see Experts registered at this site (if any).

Instructions for Experts

Please use the contact information below, if you are interested in the position or work being advertised.

You can also place your information on the Experts page.

The List

Name of offerer (Company)	Position description	Location	Contact Info	Expires date
Embedded Bits	Looking for talented embedded Linux engineers to work on our customers' real-world projects. Projects mostly entail getting Linux to work on new custom hardware and optimizing it. Visit Job Description	UK (Bristol/Cribbs)	amurray@embedded-bits.co.uk	July 2015

Category:

- Linux

> From: eLinux.org

Board and Chip Vendors

This page has a list of the companies or organizations that make processors or boards for embedded products. If you are looking for companies who sell Linux software or Linux-related services, see the Vendors page. If you are looking for companies who sell end-user products based on Linux, see Companies. If you are looking for information about specific development boards, see Category:Development Boards

Contents

A

- Allwinner Technology - An arm soc vendor based in Zhuhai, China
  - make low cost arm soc for consumer electronic products, such as A1x.
- AMCC - Applied Micro Circuits Corporation
  - makes embedded PowerPC processors??
- AMD
  - acquired ATI and now produces MIPS-based embedded processors (Xilleon series)
  - Also, used to make Geode chip
- Analog Devices - Wikipedia entry
  - ADI designs and manufactures the Blackfin processor, which has been in the mainline Linux kernel since 2.6.22 (May 2007).
  - There is a central open source site with a dedicated documentation wiki
  - Many more Blackfin manufactures can be found in that wiki as well
- ARM - Wikipedia entry
  - ARM designs ARM architecture processors, and license the technology to companies that actually make the chips, they also make a few reference boards for their own technology named ARM RealView, ARM Versatile and ARM Integrator Info
- Armadeus
  - affordable ARM boards (Freescale i.MX + Xilinx Spartan FPGA)
  - Armadeus started as a community project
- Atmel : Atmel.com - Wikipedia entry

- Atmel makes the SAMA5D3 Xplained 32-bit ARM Cortex-A5 development platform, and the older AT91SAM ARM926ej-s development platform
- AT91 Linux gateway
- Github for Linux-based systems : linux4sam.org github
- AT91SAM and Atmel MCU community forum
- Armkits Embest
  - Atmel AT91SAM9G45 Single Board Computer English Website，China Website

B

- Bluetechnix makes tiny Blackfin modules to simplify custom board development
- Boundary Devices
  - manufacturers of ARM based Single Board Computers and System on Modules
  - does custom board and device work.
- Broadcom - makes ARM chips for mobile phone market

C

- CALAO Systems sells tiny and cheap ARM based boards
- Cambridge Signal Processing makes tiny Blackfin modules to simplify custom board development
- C Data Solutions makes tiny Blackfin modules to simplify custom board development
- Cirrus Logic - Wikipedia entry
  - Cirrus Logic makes ep93xx and ep73xx 32-bit ARM-based Microcontrollers

D

- DigiKey sells the OMAP3 based beagleboard, Pandaboard and Beaglebone

E

- EFlag Tech does custom Blackfin platform designs (software and hardware)
- eInfochips - The Solution People Open-RD based platform
- electronics.cat Open source hardware. Fastest and tidiest way prototyping electronics. It can be connected to Raspberry Pi 3Bpi
- Elphel, Inc provide high performance Network Cameras based on Free Software and Hardware designs. Axis EtraxFS & Spartan 3e 1200k gates FPGA.
- Embest sells Atmel SAM9G45 based SBC6845, TI OMAP3530 based DevKit8000, DM3730 based DevKit8500D, AM3359 based DevKit8600, AM1808 based SBC8018, Freescale i.MX 6Quad based SABRE Lite, NXP LPC1788 based SBC1788 and Discover-MO:) modules for STF32F4DISCOVERY board.

F

- Freescale Semiconductor - Wikipedia entry
  - ARM
    - i.MX3x series based on the ARM11 processor
    - i.MX5x series based on the Cortex-A8
    - i.MX6x series based on the Cortex-A9
  - PPC
    - Freescale makes several PPC-based processors (and associated development boards) as well

- Forlinx Embedded - mainly focus on the manufacture and sales of ARM series development boards.
  -
    - FL2416 based on Samsung ARM9 S3C2416X http://www.forlinx.net/?p=28&a=view&r=104
    - OK6410-A based on Samsung ARM11 S3C6410 http://www.forlinx.net/?p=27&a=view&r=49
    - OK6410-B based on Samsung ARM11 S3C6410 http://www.forlinx.net/?p=27&a=view&r=50
    - OK210 based on Samsung Cortex-A8 S5PV210 http://www.forlinx.net/?p=26&a=view&r=47
    - OK210-A based on Samsung Cortex-A8 S5PV210 http://www.forlinx.net/?p=26&a=view&r=48
    - OK335xD based on TI Sitara AM335x http://www.forlinx.net/?p=26&a=view&r=46
    - OK335xS based on TI Sitara AM335x http://www.forlinx.net/?p=26&a=view&r=99
    - OK335xS-II based on TI Sitara AM335x http://www.forlinx.net/?p=26&a=view&r=110

G

- Gumstix sells various very small processor and add-on boards. Especially interesting for robotics related projects.

H

- HITEG LTD does custom embedded board ,it focused on IT outsourcing and embedded technology(software and hardware)
- Hua Heng Tech does custom Blackfin platform designs (software and hardware)
- HV Sistemas S.L. makes tiny Blackfin modules to simplify custom board development

I

- IBM - Wikipedia entry
  - Makes embedded PowerPC processor chips, such as the PPC 440 line

K

- KwikByte sells full-featured ARM based boards:
  - Such as the: KB9202, KB9260 and KBAT9261

M

- Marvell sells a lot of ARM chips.
  - One of the most interesting ones is probably the Marvell 88W8618 which is used in the Freecom MusicPal.
- MYIR sells various ARM development boards, CPU modules and single board computers.
  - MYD-AM335X TI AM335x Development Board using MYC-AM335X SoM
  - MYD-AM335X-Y TI AM335x Development Board using MCC-AM335X-Y SoM
  - MYD-AM335X-J TI AM335x Development Board using MCC-AM335X-J SoM
  - MYD-IMX28X Freescale i.MX28 Development Board using MYC-IMX28X SoM
  - MYD-SAMA5D3X Atmel SAMA5D3 Development Board using MYC-SAMA5D3X SoM
  - MYD-SAMA5D3X-C Atmel SAMA5D3 Development Board using MCC-SAMA5D3X-C SoM
  - MYD-SAM9X5 Atmel SAM9X5 Development Board using MYC-SAM9X5 SoM
  - MYD-SAM9X5-V2 Atmel SAM9X5 Development Board using MYC-SAM9X5-V2 SoM
  - Z-turn Board Xilinx Zynq-7010, 7020 Single Board Computer
  - Rico Board TI AM437x Single Board Computer

N

- NEC - makes ARM chips, used to make lots of MIPS chips
- NVIDIA - Wikipedia entry
  - NVIDIA designs and manufactures both GeForce GPUs and Tegra SoCs.
  - The Jetson TK1 board contains a Tegra SoC.

Q

- Qualcomm - makes multicore ARM MSM products that support Linux
  - Their Snapdragon platform provides a 1GHz ARM core and advanced DSP.

R

- Renesas Technology - Wikipedia entry
  - Renesas makes the SuperH, M32R, and H8 RISC CPUs, the RX CISC CPUs, and others.

S

- Samsung
  - Samsung Electronics makes the S3C24XX(ARM 9), S3C64XX(ARM 11), S5P64XX(ARM 11), S5PC100(Cortex A8), S5PV210/S5PC110(Cortex A8), EXYNOS4210(Cortex A9) SoCs and others.
  - See more Samsung Application Processors
- SiRF - Makes besides GPS products also ARM based SoC for location aware devices, like the ARM9 based Atlas III (codename) at4x0a and the ARM11 based SiRFPrima.
- SolidRun i.MX6 uSoM's, HummingBoard baseboards and i.MX6 CuBox media box
- ST-Ericsson make mobile platforms for handset form factors and similar, typically ARM based with cellular modems.
- Sunplus is a company specialising in chips for multimedia and mobile applications. One of their interesting chips is the Sunplus SPMP3050A which is used in MP4 players.
- Surveyor Corporation makes Blackfin based robot modules for academics, hobbyists, and professionals

T

- [Tianyeit]
  - Sell the tiny Package -- Computer In Package base on TI OMAP35x/AM3517/omap4430/omap4460/dm3730/am3359
- Timll
  - Reference Design and customized HW/SW Design Based on TI OMAP35x/AM3517
- TechNexion
  - Expandable and fully customizable ARM system modules and interface boards
  - Embedded x86 boards with coreboot and linux core extensions
- Texas Instruments - Wikipedia entry
  - Texas Instruments makes the MSP430 MCUs, TMS320 C2000/C5000/C6000 DSPs, DaVinci/OMAP ARM+DSP-based processors, and others.

V

- VIA Embedded
  - ARM boards based on VIA and Freescale SoCs, such as the VAB-600 Springboard, VAB-800/820/1000, the APC

series 8750/Rock/Paper

- small form factor x86 boards for embedded, EPIA-900/910 series.

Category:

- Hardware

eLinux.org Information and Usage tips

General information about this site, and related resources.

- Homepage
- Twitter Follow

Help:About

This is the embedded Linux wiki.

The purpose of this wiki is to preserve and present information about using Linux in embedded systems. The primary target audience is developers, engineers, managers, hobbyists and others who are involved with putting Linux into embedded systems.

Please utilize this information in your own development efforts, and please provide back information when you can. See Volunteer editor tasks to get started, if you don't know where to begin.

Sponsors

This site is sponsored by the Linux Foundation, with OSUOSL providing Hosting. Bill Traynor is the primary administrator and maintainer of this site.

History

This site was created in late 2006 and early 2007, and opened to the public in June 2007. The major initial content for this site came from merging content from two previous sites:

- the original elinux.org site
- the CELF public wiki

The original elinux.org site had a lot of content about putting Linux onto existing products (often, products which were not shipped with Linux or designed to support it).

The CELF public wiki has a collection of resources, including research results, collections of links to interesting articles, and many presentations from CELF and other conferences.

In 2006, it was decided to create a larger site with a combination of the two sets of resources. CELF hired a part-time administrator to work on this, and Movial provided hosting for the site.

Originally the site was hosted by Movial

In 2008, the site was moved to a virtual host, hosted by GoDaddy.com.

Some concepts for the site came out a CELF AG meeting in 2006. Notes from that meeting are at Embedded Wiki Design. A task force was formed, which held a conference call, with notes at: Embedded_Wiki_Task_Force_Conference_Call

Category:

- Community

From: eLinux.org

Help:Contents



Important note: When you edit this page, you agree to release your contribution into the public domain. If you don't want this or can't do this because of license restrictions, please don't edit. This page is one of the Public Domain Help Pages, which can be freely copied into fresh wiki installations and/or distributed with MediaWiki software; see **Help:Contents** for an overview of all pages. See Project:PD help/Copying for instructions.



Reading

- Navigation
- Searching
- Tracking changes
- Watchlist

Editing

- Editing pages
- Starting a new page
- Formatting
- Links
- User pages
- Talk pages
- Signatures

Advanced editing

- Images
- Tables
- Categories
- Subpages
- Managing files
- Moving (renaming) a page
- Redirects
- Deleting a page
- Protected pages
- Templates
- Variables
- Namespaces
- Special pages
- External searches
- Bots

Personal customization

- Preferences
- Skins

Wiki administration

- Sysops and permissions

The following features require extra permissions that are not normally granted to all wiki users.

- Protecting and unprotecting pages
- Sysop deleting and undeleting
- Patrolled edits
- Blocking users
- Range IP blocks
- Assigning permissions

Language:

English

Category:

- Help

From: eLinux.org

Help:Editing

	Important note: When you edit this page, you agree to release your contribution into the public domain. If you don't want this or can't do this because of license restrictions, please don't edit. This page is one of the Public Domain Help Pages, which can be freely copied into fresh wiki installations and/or distributed with MediaWiki software; see Help:Contents for an overview of all pages. See Project:PD help/Copying for instructions.	

Language:

English

Category:

- Help

> From: eLinux.org

ELinuxWiki:Mailing List

Contents

eLinux.org Lists

If you want to discuss this wiki with its primary maintainers and editors, please join the appropriate list below:

elinux-discuss@lists.elinux.org

- Information and archives for **discuss**

This list is for general discussions around Embedded Linux. **elinux-discuss@lists.elinux.org**

To filter this mailing list, use the following: List-Id: \

---

elinux-announce@lists.elinux.org

- Information and archives for **announce**

This list is reserved for occasional announcements. It is read only. **elinux-announce@lists.elinux.org**

To filter this mailing list, use the following: List-Id: `<elinux-announce.lists.elinux.org>`

---

elinux-wiki-dev@lists.elinux.org

- Information and archives for **elinux-wiki-dev**

This list is for general discussion of development of the elinux.org wiki. **elinux-wiki-dev@lists.elinux.org**

To filter this mailing list, use the following: List-Id: `<elinux-wiki-dev.lists.elinux.org>`

Project Lists

- Information and archives for the **minnowboard**

This list is for discussion of the Minnowboard development board. **elinux-minnowboard@lists.elinux.org**

To filter this mailing list, use the following: List-Id: `<elinux-minnowboard.lists.elinux.org>`

External Embedded Linux Lists

- Linux Kernel - Embedded Linux
  - Archive
- CE Linux Developer List (sponsored by CE Workgroup of the Linux Foundation)
  - Archives

Deprecated discussion list

The folowing list is deprecated

- elinux-wiki
- elinux-wiki Archive

Category:

- Community

From: eLinux.org

ELinuxWiki:Irc

IRC Channels

Freenode irc.freenode.net

#eLinux	IRC for the eLinux.org community (Archives). (Web-based access via Mibbit)
#edev	Discussion for small embedded device development and hacking. (Archives)
#openezx	Discussion about Motorola's EZX phone platform, such as the A780, E680 and E680i Linux based GSM phones. (Archives)
#dlinkhacking	This is the place to discuss software and hardware hacking of ALL DLink products
#linux-sh	Linux on SuperH(SH) Devices.
#mipslinux	Linux on MIPS Devices.
##electronics	General electronics discussion.
#oe	Channel for discussion of OpenEmbedded development environment OE Homepage. Archives Old archives
#davinci	Discussions regarding the TI Davinci platform and EVM related hardware.
#openmoko	Open Source Telephony; (Archives)
#maemo	Nokia Internet Tablet OS; (Archives, Alternate archives)
#maemo-alternatives	Discussions about getting the latest Linux kernels and distributions (principally ArchLinux) to work on the Nokia N900
#neuros	Neuros OSD Hacking
#usdtv	USDTV/Hisense HDTV Receiver Hacking
#u-boot	U-boot bootloader
#openjtag	where the openocd folks hang out. (Archives)
#fedora-arm	Fedora on ARM based processors
#ci20	MIPS Creator CI20 related discussions
#beagle	BeagleBoard related discussions (Archives, Web based interface)
#tuxdroid	Discussions regarding the Tux Droid
#nslu2-general	NSLU2 end user channel.
#nslu2-linux	NSLU2 developer channel. Archives
#raspberrypi	25$ Raspberry Pi computer board channel
#qi-hardware	Qi-hardware initiative to "OpenSource" the hardware side. Qi-hardware

OFTC irc.debian.org

#emdebian	Debian-based embedded and crosstools stuff, and emdebian distribution
#debian-arm	Debian ARM port

Category:

- Community

> From: eLinux.org

Wanted

This article lists pages that are requested to be added to this wiki. The page specifically lists pages that have been requested by users. (Another way to do this is to create a stub page for the desired page, and add the tag for Category:NeedsEditing.) However, this page lists pages that people would like to see, but have insufficient knowledge or expertise to create.

An automatically generated list of pages that are referenced, but do not exist, is at Special:Wantedpages

User-requested pages

Put your request for a page or article you would like to see on this wiki in the list below:

Realtime testing best practices I'd like to see a document collecting existing information about realtime testing, with links to test programs and information about already-performed tests (with pros and cons for each approach). TimBird

comparison of different C libraries

Kernel debugging processors from user space (kgdb and other tools) Dinesh

How to select a kernel version What are the steps to take or the factors to consider when picking a kernel version? Given a CPU family, to what extent has support been added to the mainline sources on kernel.org, or must a more architecture-specific source be used? Pointers to high-level feature or driver support (like the release summaries on kernelnewbies.org) might be useful too.

Category:

- CELinux Wiki

Technology Watch List

This page lists technologies and projects that CELF members are interested in the status of. This includes kernel patches, new technology research, and middleware and user-space projects of key interest for consumer electronics products. The projects may be the topics of discussion at CELF meetings, and we plan to watch and report the status of these technologies.

Please add any information you have about the technology items listed below!!

Contents

Latest Watchlist

The **Status** field in the table below indicates whether this feature is on track for being mainlined. The **When was last activity** field indicates the kernel version number or date when the last activity was noted for this feature. This could be the last kernel version where bits from this patch were mainlined, or the last date of visible feature development activity outside the main tree.

See also: Embedded linux status

Kernel Stuff

Size Stuff

Technology, Feature or Patch	Status	When was last activity	Notes
Linux-tiny	In active maintenance. Path set published for 2.6.24	Latest full patchset was published Oct. 13, 2007, Latest patch (CONSOLE_TRANSLATIONS) was mainlined in June 2008	Maintainer is Michael Opdenacker (with help from Thomas Petazzoni). See Linux Tiny Patch Details for details about the patch status.
kpagemap - memory instrumentation	mainlined in Feb, 2008 (for 2.6.25)	Feb 2008	• Can show details about every allocated and virtual page on the system. • Introduces PSS and USS size metrics ◦ PSS - Proportional Set Size ◦ USS - Unique Set Size • Resources: ◦ ELC 2007 presentation: http://selenic.com/repo/pagemap/raw-file/tip/memory-profiling.html ◦ LWN.net article: http://lwn.net/Articles/230975/ ◦ Visualization tools: http://selenic.com/repo/pagemap
Bloatwatch (2.0)	Now actively reporting kernel sizes	April 2008	See Matt's presentation from ELC 2008
gcc -ffunction-sections -fdata-sections	Patches submitted to LKML	July 2008	See Function sections

File Systems

Technology, Feature or Patch	Status	When was last activity	Notes
Squash Fs	Mainlined in January 2009 (for 2.6.29) See http://lwn.net/Articles/314326/.	As of Feb, 2009, Phillip was working on some user-space tools issues, to support the new filesystem format (version 4.0)	Good job Phillip!
AXFS	Not mainlined.	Some required (predecessor) bits were mainlined in March, 2008, A version was submitted to LKML in August 2008	Flash filesystem that allows for tuning the amount of XIP • See AXFS: Architecture and Results - Jared Hulbert's presentation for ELC 2008
LogFS	Not mainlined.	Last mainline attempt was May, 2008.	• Home page is at: http://logfs.org/logfs/ • Jörn Engel is working on a re-write to address several major issues. • CELF is funding this work.
UBIFS	mainlined in 2.6.27		Resources: • home page? • white paper • LWN.net article
Pram Fs	submitted for review for 2.6.31	June 2009	See http://lkml.org/lkml/2009/6/13/86

Tracing and instrumentation

Technology, Feature or Patch	Status	When was last activity	Notes
LTTng	core not mainlined. Markers were mainlined in 2.6.24	?	LTTng instrumentation was changed to use markers, in early 2007
SystemTap (and Kprobes) for non-i386 arches	ARM support merged for 2.6.25	?	KProbes ports for ARM, MIPS and PPC32 were reported on at ELC 2007, SystemTap for SH was demo'ed at ELC 2007
Kernel Function Trace (KFT)	not mainlined	Last published patches for 2.6.22	Nicholas McGuire was taking over maintainership from Tim Bird, with funding from CELF. Haven't heard much recently (as of June 2008)
ftrace	mainlined in 2.6.27		This was formerly the latency-trace features from the RT-preempt patch set, refactored for general use
printk-times (arch support)	fully mainlined?	April, 2005	Some arches had problems with accessing the clock too early in the kernel bootup sequence, but a new setup routine defers turning on the timestamping until after timekeeping is initialized

Realtime

Technology, Feature or Patch	Status	When was last activity	Notes
KTimers	mainlined, but needs lots of porting to embedded architectures	2.6.23 / 2.6.24	Adding clock driver support for various architectures is an ongoing process
RT-preempt	some parts mainlined	current RT patches are still against 2.6.26	Next target is to integrate threaded interrupts in 2.6.29 - as discussed on LinuxPlumbers conference Oregon 2008
Xenomai	external project	2.6.25 - stable release, newer in development	Xenomai is a real-time development framework cooperating with the Linux kernel, in order to provide a pervasive, interface-agnostic, hard real-time support to user-space applications, seamlessly integrated into the GNU/Linux environment. Ready to deploy.

Security

Technology, Feature or Patch	Status	When was last activity	Notes
App Armour	not mainlined	May, 2007	• LSM framework was removed from kernel in 2.6.24 • AppArmour group was let go from Novell in late 2007 ◦ See http://www.news.com/8301-13580_3-9796140-39.html ◦ LWN.net article
TOMOYO Linux	not mainlined	Nov 17, 2007 (4th post) (trying now)	"TOMOYO Linux has only recently surfaced on the wider mailing lists; its reception has not been entirely friendly. This project's developers have some work to do if they are (1) to get past the same obstacles which have slowed AppArmor, and (2) show that their project is sufficiently different from AppArmor to merit inclusion as yet another security framework." (from Linux Weather Forecast)
SMACK	mainlined in 2.6.25	.	LWN.net article on SMACK
SELinux (usable in embedded)	SELinux is mainlined, but some issues for making SELinux usable for embedded remain	April 2008	• There has been much progress recently to support SELinux in the embedded space • Requires filesytem with xattrs (some flash filesystems do not support xattrs) • at ELC 2008, Yuichi Nakamura described an embedded configuration of SELinux in as little as 700K ◦ Development of Embedded SELinux - Yuichi Nakamura presentation at ELC 2008

Power Management

Technology, Feature or Patch	Status	When was last activity	Notes
powertop	support for powertop is mainlined (for x86 architecture)	?	• Powertop shows timers and power state durations • Only well-supported on x86?? ◦ To support on other architectures, the platform needs to support CPUIdle interface, in order to show C-state (power state) ** There has been some activity for non-Intel processors
PM QoS	in 2.6.23-mm1	Oct '07	(see http://lesswatts.org) need Embedded folks to take a look and help define the interface, expand the features and raise issues from the embedded perspective.
Wolfson voltage regulator stuff	not mainlined??	March 2008?	See Every Microamp is Sacred - A Dynamic Voltage and Current Control Interface for the Linux Kernel - Liam Girdwood's ELC 2008 presentation

Bootup Time

Technology, Feature or Patch	Status	When was last activity	Notes
deferred module load	not mainlined, no patches	July 2008	Proposed, without patches, by Tim Bird on linux-embedded list
async initcalls	not mainlined	July 2008	Patches submitted by Arjan van de Ven
snapshot boot	not mainlined	July 2006	Sony presented paper at OLS 2006. This technology is used in Sony products.
fastboot kernel configuration option	not mainlined?	July 2006	Arjan van de Ven submitted as part of his async initcalls patches

Miscellaneous

Technology, Feature or Patch	Status	When was last activity	Notes
Userspace I/O	Seems to be merged into mainline (see: http://lwn.net/Articles/242483/)	July, 2007	References: http://www.kroah.com/log/linux/uio.html

Middleware

Project	Status	When was last activity	Notes
libdlna	Developer has added support for all profiles except MPEG-4 and WMV (http://hg.geexbox.org/libdlna/)	29 Aug, 07	Short term goal is to provide DLNA support to Ushare media server, long term goal is to provide generic DLNA reference library References: http://libdlna.geexbox.org/

Category:

- Community